Erwin Neutzsky-Wulff

AMSTRAD BASIC

AMSTRAD

CPC 464

Borgen



Amstrad BASIC

Af samme forfatter:

Dialog, roman 1971
Udstillingsbilleder, digte 1972
Tidsmaskinen, essay 1972
Seks hundrede seksogtres, digte 1972
Adam Harts opdagelser, roman 1972
Kærlighed, digte 1972
Etik, essay 1973
Parentes begynd, digte 1973
Anno Domini, roman 1975
Gud, roman 1976
Victor Janis og søn, roman 1977
Adam Hart og sjælemaskinen, roman 1977
Oiufael, roman 1977
Den treogtredvte marts, roman 1977
Havet, roman 1978
Indsigtens sted, roman 1980
Menneske, roman 1982
Mikrodatamaten. Programmering og Anvendelse, en bog om ZX 81 BASIC, 1982
Programmering med Commodore BASIC, 1983
BASIC-programmering med Texas 99/4A, 1983
BASIC med Commodore 64, 1984
Okkultisme, 1985

ERWIN NEUTZSKY-WULFF

Amstrad BASIC

BORGEN

Omslag: Lars Beierholm
Trykt hos Narayana Press
ISBN 87-418-7530-3

Til min elskede hustru
Elsebeth

INDHOLD

1. VARIABEL OG PROGRAM

PRINT – CLEAR – RUN – LIST – NEW – CONT – EDIT – GOTO

Du har måske aldrig set en datamat før (det regner denne bog sådan set med, at du ikke har), men en lommeregner har du sikkert haft i hånden. Du ved måske også, at nogle af de lidt dyrere modeller har "hukommelse".

Det har en datamat – ikke overraskende – også. Lad os prøve at få den til at huske et tal, f.eks. 1234. At lade maskinen huske et tal for os er at lægge det ned i en skuffe med et genkendelsestegn udenpå. Skuffen hedder en *variabel* (det, der er i, varierer – vi kan smide dens indhold ud og komme noget andet i), og vi genkender skuffen ved hjælp af dens *variabelnavn*. Vi vil nu prøve at give maskinen en ordre. Skriv

```
a = 1234
```

Læg mærke til, at lighedstegnet står øverst på tasten. Vi får det ligesom på en almindelig skrivemaskine ved at holde skiftenøglen (SHIFT – der er to af dem, og det er lige meget, hvilken du bruger) nede.

Det, vi skrev, kom til at stå lige under ordet Ready, der betyder "parat". Når som helst du ser dette ord, ved du, at maskinen ikke i øjeblikket har andet at lave end at modtage dine ordrer. Det ser for øjeblikket nærmest sådan ud på skærmen (fraregnet firmanavn, osv.):

```
Ready
a = 1234
```

Der er også en lille sort klods (gul, hvis du har farvemonitor) efter 4-tallet. Det er en *markør*. Den følger dig, når du skriver, eller rettere sagt: Den er altid lige der, hvor det næste bogstav, tal eller tegn (den næste *karakter*) kommer til at stå. Den fortæller dig med andre ord, hvor du er henne på skærmen.

Hvad er det nu overhovedet, vi har siddet og sagt til maskinen? Jo, sådan noget som, at a skal være 1234. Vi har lagt tallet 1234 ned i skuffen a.

Næsten. For vi har sådan set ikke fortalt maskinen, at vi er færdige med at »bestille«. Det kunne jo være, vi ville skrive 12345 eller 12345667788.

Vi signalerer til maskinen, at vores ordre er færdig, med tasten ENTER, der er stor og blå og ikke til at tage fejl af. Der er en lille en nederst til højre, og den kan vi også bruge.

Du opdager måske nu, at også tallene står to steder, nemlig over bogstaverne som på en almindelig skrivemaskine og ude til højre arrangeret som på en regnemaskine. Dette skyldes, at nogle er meget dygtige til at betjene regnemaskiner og har svært ved at omstille sig til at "regne på skrivemaskine".

Om du vil bruge det ene eller andet 7-tal, er ganske op til dig. Resultatet bliver det samme.

Det var ENTER, vi kom fra. Tryk på tasten. Der står nu

```
Ready
a = 1234
Ready
```

Maskinen har udført ordren (det var den jo ikke så længe om) og venter på en ny. Nu kunne den jo sådan set godt snyde os, for vi kan ikke se skuffen. Derimod kan vi bede om at få opgivet dens indhold. Skriv

```
print a
```

Det betyder: Skriv (indholdet af skuffen) a, og det gør den så: 1234, fulgt af et nyt Ready. Nu ved vi, hvordan vi får maskinen til at huske, og det var såmænd ikke så svært.

Vi taler også om at *tilskrive en varibel en værdi.* I dette tilfælde har vi tilskrevet (variablen) a værdien 1234.

a er som sagt variabel*navnet.* Vi kunne også have kaldt den peter eller jens. Prøv

```
peter = 12
```

Det kunne jo f.eks. betyde, at Peter er tolv år. Måske synes du, navnet skal med stort, altså Peter, men det gør faktisk ingen som helst forskel. Prøv

```
print peter
```

```
print Peter
```

og

```
print PETER
```

eller simpelt hen

```
print peter ; Peter ; PETER
```

Nemlig, de er alle sammen 12. For maskinen er de tre måder at skrive det samme variabelnavn på.

Læg også mærke til semikolonnet i det sidste eksempel. Det gjorde, at vi kunne nøjes med en enkelt print-ordre. Vi kunne jo nemlig ikke så godt skrive

print peterPeterPETER

Det ville maskinen have opfattet som een variabel. Prøv engang. peterPeterPETER har værdien 0 som alle variabler, som endnu ikke har fået tilskrevet værdi (ø for ikke at forveksle det med bogstavet o). Hvad med

print peter Peter PETER

Jo, den er også god nok. Mellemrummet fortæller maskinen, at der er tale om tre forskellige variabler.

Dette betyder selvfølgelig omvendt, at der ikke kan være mellemrum i et variabelnavn. Derimod er alle bogstaver og tal i orden, blot den første karakter i variabelnavnet er et bogstav.

Jo, så må der heller ikke være mere end 40 karakterer. Hvad nu, hvis vi vil kalde en variabel for print? Prøv

print = 12

Maskinen siger

Syntax error

Syntaxen er reglerne for, hvad man må og ikke må sige i sproget BASIC. Og print er "reserveret" som ordre, det kan ikke bruges som variabelnavn, for ikke at forvirre maskinen.

I nogle dialekter af BASIC må et variabelnavn ikke engang *indeholde* et reserveret ord. "Sprinter" som variabelnavn går altså heller ikke. Dette er helt i orden i AMSTRAD BASIC. Prøv bare

sprinter = 12

Til gengæld må vi så selvfølgelig sætte mellemrum omkring de reserverede ord.

printa

går ikke, eftersom maskinen vil opfatte det som navnet på en variabel.

Kan maskinen kun huske tal? Nejda. Prøv

a $ = "amstrad"

og

print a $

Læg mærke til gåseøjnene og dollartegnet. Et ord eller en bogstavsammenstilling (som også kan indeholde tal og tegn) kaldes en *streng*, og strenge skal i gåseøjne. Hvis det ikke var for gåseøjnene, kunne amstrad jo lige så godt være et variabelnavn.

En variabel, der gemmer på en streng, kaldes endvidere en *strengvariabel*, og at der er tale om en strengvariabel, ser maskinen af dollartegnet i enden på variabelnavnet. Omvendt kaldes en variabel, der gemmer på et tal, for en *numerisk* variabel. Skriv

print a a $

Maskinen skriver

1234 amstrad

Prøv nu

print 123

Ja, maskinen skriver simpelt hen 123. Vi kan også regne på den måde. Skriv

```
print 123 + 456
```

eller blot

```
? 123 + 456
```

hvor spørgsmålstegnet er en "forkortelse" for print. Maskinen regner det ud og svarer 579. Prøv på samme måde 456 – 123 (minus), 123*456, 123/456 (divideret med), osv. Du kan altid regne med amstrad. Prøv endelig

```
print a a$ 123
```

Nej, nu bliver maskinen forvirret. Gør dig det til en regel for en sikkerheds skyld altid at bruge semikolon.

```
print a ; a$; 123
```

er i al fald i orden. Prøv komma i stedet for, altså

```
print a , a$, 123
```

Vi får "kolonner". Prøv

```
print 1 , 2 , 3 , 4 , 5 , 6
```

så ser du, hvordan det fungerer.

Endelig er der en særlig slags variabler, der hedder *heltalsvariabler.*

Tal, der skal gemmes, optager selvfølgelig plads. Du kan sikkert forstå, at et tal som 1 . 23456789 optager mere plads end 1 eller 12. Derfor har man også af pladshensyn

nogle særlige små skuffer, som kun kan indeholde hele tal. En heltalsvariabel angives ved % svarende til $ for strengvariablerne. Prøv

```
a%=1.23456789
```

og

```
print a%
```

Ja, maskinen har "glemt" decimalerne, for der var ikke plads til dem i den type skuffe, vi valgte. Det går selvfølgelig lige så galt, hvis vi prøver at presse et for "højt" tal ned i skuffen. Prøv

```
a%=32767
```

og derefter

```
a%=32768
```

I det sidste tilfælde siger maskinen

```
Overflow
```

skuffen "flyder over". Den højeste værdi, en heltalsvariabel kan tilskrives, er altså 32767.

Tilsvarende er den længste streng, der kan være i en strengvariabel, på 255 karakterer. Forsøger vi at tilskrive en længere, får vi

```
String too long
```

Alle disse bemærkninger fra maskinens side kaldes *fejlmeddelelser*. Den strejker ikke bare eller bliver fornærmet, når

der er noget, den ikke kan, den fortæller os hvorfor. Den almindeligste fejlmeddelelse er naturligvis Syntax error, som maskinen bruger, når vi har skrevet forkert.

Almindelige numeriske variabler har selvfølgelig også deres begrænsning, men den er temmelig astronomisk, cirka svarende til et 1-tal med 38 nuller efter!

Vi kan altid blive af med indholdet af en skuffe ved at komme noget nyt i den. Prøv

```
print a
```

og så

```
a = 37
```

og endelig igen

```
print a
```

Vi kan også på een gang tømme alle skuffer. Skriv

```
clear
```

Nu har maskinen glemt alt om a , a$, osv. Prøv selv. Læg mærke til, at maskinen skriver indholdet af en tom strengvariabelskuffe som en tom linje.

Maskinen "glemmer" selvfølgelig også alt, hvis du slukker for maskinen. Det er derfor, du skal lære at gemme ting på bånd.

En lidt anderledes værditilskrivning er

```
a = a + 1
```

Prøv

```
a = 987
```

og

```
print a
```

derefter

```
a = a + 1
```

og atter

```
print a
```

A er nu 988, for ordren a = a + 1 betød ”gør a 1 større”. Prøv tilsvarende a = a * 2 (gang a med 2). Vi kan gøre noget lignende med strenge. Prøv

```
a$ = "am"
```

og

```
a$ = a$ + "strad"
```

og

```
print a$
```

Her vil andre ”regnetegn” end + være meningsløse for datamaten. Du kan altså ikke sige a$ – ”strad” eller lignende.

Måske har du skrevet forkert et par gange. Du har så formodentlig klaret dig med at trykke på ENTER, tage din Syntax error og prøve igen. Du kan imidlertid gøre det lettere. Skriv

```
pra
```

Du vil nu gerne af med det forkerte a. Det ordner du med tasten DEL (DELete = slet).

Du forstår sikkert, at variabler er vigtige i programmering. Nu ved du så noget nær, hvad der er at vide om dem, og vi fik også samlet et par andre nyttige ting op undervejs. Lad os repetere lidt. Skriv

```
a = 90

a = a + 10

print a

clear

print a
```

Disse fem ordrer demonstrerer lidt om variabler. Læg mærke til, at vi skal taste dem ind i den rigtige rækkefølge. Hvorfor ville det ikke nytte noget med

```
print a

a = 90

clear

a = a +1

print a
```

Ja, blandt andet kunne vi godt droppe a = 90, når vi sletter værdien umiddelbart efter. Det er også besværligt, hver

gang vi skal demonstrere, at taste alle ordrerne ind. Faktisk kan vi nøjes med at gøre det en enkelt gang, hvis vi laver et *program*. Det kunne se sådan ud:

```
10 a = 90
20 a = a + 10
30 print a
40 clear
50 print a
```

Som du ser, har vi simpelt hen nummereret ordrerne, de er blevet *programlinjer* med *linjenumre*. Som regel nummererer man 10-20-30, osv. Hvorfor? Jo, for så kan man senere passe noget ind, man har glemt i første omgang.

Sådan et program kan man "køre", og så vil maskinen automatisk udføre alle ordrerne i nummerorden. Først skal vi selvfølgelig have skrevet programmet ind i maskinen. Lad os prøve. Skriv

```
10 a = 90
```

Tast nu ENTER. Normalt ville maskinen udføre ordren og skrive Ready, men linjenummeret fortæller den, at vi er ved at skrive et program, "gemme" en række ordre til senere udførelse.

Skriv på samme måde linje 20-50. Hvordan kører vi nu programmet? Det gør vi med ordren

```
RUN
```

Tast ENTER, og alle ordrerne bliver udført i den rigtige rækkefølge. Først bliver a 90, så bliver den 10 højere, altså 100, så skriver maskinen de 100, sletter værdien og skriver den nye værdi, 0.

Skriv RUN igen, og det samme gentager sig. Læg mærke

til, at dit program nu står omtrent midt på skærmen. Hver gang du skriver noget nyt, rykker det allerede skrevne en tand op, indtil det simpelt hen forsvinder op over kanten.

Prøv at køre programmet, til program*listen* er helt væk. Er dit program slettet nu? Slet ikke. Du kan stadig køre det med RUN, og du kan også få det at se, nemlig med

```
LIST
```

Maskinen har imidlertid rettet lidt i dit program, den har for overskuelighedens skyld skrevet alle reserverede ord med stort, så det nu ser sådan ud:

```
10 a = 90
20 a = a + 10
30 PRINT a
40 CLEAR
50 PRINT a
```

Lad os prøve at skrive et andet program. Først må vi selvfølgelig af med det, som allerede står i maskinen, og det gør vi med

```
NEW
```

Prøv så at skrive programmet

```
10 PRINT "AMSTRAD"
20 RUN
```

Maskinen bliver ved og ved med at skrive AMSTRAD. Hvorfor nu i alverden det? Jo, for så snart den kommer ned i linje 20, støder den på RUN, der er en ordre om at køre hele programmet forfra igen, og så fremdeles.

Sådan et program eller eventuelt en del af et program,

hvor maskinen ”kører i ring”, kaldes en *løkke*, og i vores tilfælde er det til og med en *uendelig løkke*. Maskinen kan simpelt hen ikke ”slippe ud”.

Det kunne altså se ud, som om vi var nødt til at slukke og tænde igen for at ”undslippe”. ESC øverst til venstre på tastaturet betyder imidlertid netop ESCape = undslippe. Ved at trykke på den, kan vi altid standse en programkørsel.

Tryk en enkelt gang. Maskinen standser.

Således anvendt er ESC en slags ”pause-knap”. Tryk på en hvilken som helst *anden* tast, f.eks. ENTER, og programmet fortsætter. Brems nu atter programmet med et tryk på ESC, og tryk derefter på ESC endnu en gang. Denne gang får vi

```
Break in 20
Ready
```

Det betyder: Programmet standset i linje 20. ESC 1 gang er altså en pause, som varer, til du trykker på en anden tast, mens ESC 2 gange er endegyldig programstandsning.

Den er dog ikke mere endegyldig, end at du kan fortsætte i programmet, hvor du slap, med

```
CONT
```

Også mens du skriver et program, kan du naturligvis komme til at skrive forkert, men så er proceduren ganske som i den ”direkte ordre”. Men hvad, hvis du er ude af linjen, ja, måske færdig med hele programmet, og *så* vil rette?

Ja, så skal du have fat i den forkerte linje. Og det får du med EDIT. Stands programmet, hvis det kører, med 2 gange ESC, og skriv så

```
EDIT 10
```

EDIT betyder ”redigere”, og du har nu fået linje 10 ned til redigering. Læg mærke til, at markøren står oven på den første karakter. Du skal nu flytte den til den karakter, du vil rette.

Markøren flytter du med de små pile øverst til højre på tastaturet. For hver gang du trykker på pilen, der peger til højre, flytter markøren en karakter i denne retning. Prøv tilsvarende at flytte den til venstre.

Du kan ikke komme ud af linjen, og maskinen ”pipper”, når du har nået grænsen. Derimod kan en *program*linje godt fylde flere *skærm*linjer, og så er det, du kan få brug for pilene, der peger (og flytter markøren) op og ned. Du kan også holde tasten nede, så markøren farer af sted, til du slipper den.

Lad os nu prøve at rette AMSTRAD til MØRBRAD på den nemmeste måde. Flyt markøren hen *bag* det, der skal slettes, altså *oven på* R.

Tast DEL 4 gange. Skriv så MØRB (du kan bruge nul som stort Ø) og tast endelig ENTER. Tag en LIST. Programmet ser nu således ud:

```
10 PRINT "MØRBRAD"
20 RUN
```

Skriv

```
CONT
```

Maskinen protesterer:

```
Cannot CONTinue
```

Den kan ikke fortsætte i programmet, fordi det ikke længere er det samme program. Skriv RUN i stedet, og programmet kører igen.

Øv dig i at redigere i programmer. Det letteste er som sagt at slette det forkerte og skrive det rigtige.

I begyndelsen vil du nok få vanskeligheder og måske gøre fejlen i linjen værre. Så kan du altid komme ud af linjen med ESC. Linjen forbliver da, hvad den var, *før* du begyndte at rette i den.

Der er også andre måder at rette på. EDIT 10 og markøren på R. Tast nu CLR (tasten til venstre for DEL), og R, altså den karakter, markøren hviler på (i stedet for den foran), forsvinder. Endelig kan du bruge COPY midt i markørtasterne. Med markørtasterne (og SHIFT holdt nede) placerer du markøren i den linje, du vil rette i.

Derefter finder du stedet med COPY og retter fejlen.

En kopi af linjen med fejlen rettet vil vise sig nederst på skærmen.

Hvis du ikke kan finde ud af CLR og COPY, skal du ikke ærgre dig, for de er ganske overflødige, og den gennemgåede måde (med DEL og EDIT) er den letteste og bedste.

Måske opdager du slet ikke fejlen, men så skal maskinen nok gøre dig opmærksom på den. Skriv en ny linje 10:

```
10 PRANT "AMSTRAD"
```

Prøv at køre det. Ikke nok med, at maskinen skriver

```
Syntax error in 10
```

den giver dig linjen, ganske som hvis du havde tastet EDIT. Markør på N. DEL.

Skriv I. ENTER, og sagen er klar.

Kør programmet. Læg mærke til, at en PRINT-ordre bruger mindst en fuld skærmlinje. Her kan du imidlertid med fordel bruge komma og semikolon. Prøv både

```
10 PRINT "AMSTRAD",
```

og

```
10 PRINT "AMSTRAD";
```

Husk, at mellemrum, du vil have med på skærmen, skal stå *inden for* gåseøjne, altså f.eks.

```
10 PRINT "AMSTRAD ";
```

Når du indsætter mellemrummet, skal du selvfølgelig ikke slette noget i linjen. Anbring blot markøren efter det, du vil indsætte, altså i dette tilfælde oven på de sidste par gåseøjne, og tast mellemrum og derefter ENTER. LIST vil vise dig, at alt er, som det skal være.

Hvad nu med

```
10 a=a+1
20 PRINT a;
30 RUN
```

Maskinen begynder forfra igen, tæller 1 til, begynder forfra igen, etc. Eller gør den? Faktisk ikke.

RUN betyder nemlig ikke bare kør programmet, men kør programmet efter at have »nulstillet« alle variabler, CLEAR er så at sige "indeholdt" i RUN. Derfor bliver a heller aldrig mere end 1. Men prøv så

```
10 a=a+1
20 PRINT a;
30 GOTO 10
```

Linje 30 betyder helt bogstaveligt: Gå til 10. Og maskinen tæller nok så nydeligt. Syv-tabellen kunne vi få med

```
10 a=a+7
20 PRINT a;
30 GOTO 10
```

Vi har i dette kapitel fundet ud af nogle grundlæggende ting om, hvad en variabel og et program er. Vi har lært at skrive og redigere simple programmer, og vi har også fundet ud af, hvordan vi bruger maskinen som regnemaskine.

Prøv

```
? 123*456+789
```

Maskinen skriver ganske rigtigt 56877. Men hvad nu, hvis vi ville have haft den til at lægge 456 og 789 sammen og gange resultatet med 123? Ville

```
? 456+789*123
```

gøre det? Nej, maskinen ganger og dividerer altid, før den lægger til og trækker fra. Det kan vi imidlertid ændre på med parenteser. Vi får så

```
? (456+789)*123
```

Vi blev også præsenteret for nogle fejlmeddelelser. Her er en til. Skriv

```
10 PRINT "AMSTRAD";
20 GOTO 11
```

Maskinen skriver meget fornuftigt:

```
Line does not exist in 20
```

Den linje, der hentydes til i linje 20 (linje 11), eksisterer ikke.
Vi lærte også at bruge RUN og LIST. Skriv

```
10 PRINT 1
20 PRINT 2
30 PRINT 3
40 PRINT 4
50 PRINT 5
```

Nu kan du øve dig lidt. RUN og LIST kan nemlig også bruges i forbindelse med linjenumre.

RUN er enklest. RUN 20 betyder "Kør programmet fra og med linje 20 og ud". Med LIST er der imidlertid flere muligheder.

```
LIST 20 –
```

giver listen fra og med linje 20 og ud.

```
LIST – 30
```

giver listen til og med linje 30.

```
LIST 20 – 40
```

giver listen fra og med linje 20 til og med linje 40. Læg mærke til, at

```
LIST 20
```

giver linje 20 og ikke andet.

Endelig er det måske irriterende hele tiden at skulle holde SHIFT nede, hvis du skal skrive meget med store bogstaver. Du kan så taste CAPS LOCK (tasten til venstre for A),

og maskinen vil skrive med stort, indtil du trykker på denne tast igen.

Tast CAPS LOCK. Skriv nu AMSTRAD og derefter "!". Læg mærke til, at tegnene øverst på tasterne, som du normalt ville få med SHIFT, ikke er påvirket af CAPS LOCK, du skal stadig bruge SHIFT for at få dem.

Hold nu CTRL nede, idet du trykker på CAPS LOCK. Maskinen vil nu både give dig store bogstaver og tegnene øverst på tasterne, og du kan kun bruge tallene til højre på tastaturet. For at komme ud af denne lidt specielle situation må du igen holde CTRL nede og taste CAPS LOCK.

Vi er nu rustede til at begynde at programmere for alvor. Det gør vi i kapitel 2.

2. DE FØRSTE PROGRAMMER

USING – INPUT – DELETE – IF – THEN – ELSE – CLS

I første kapitel fandt vi ud af, hvad en variabel og et program er. Vi fik endda skrevet et rigtig nydeligt lille et, der gav os 7-tabellen. Det så sådan ud:

```
10 a=a+7
20 PRINT a;
30 GOTO 10
```

En anden metode ville være

```
10 a=a+1
20 PRINT a*7;
30 GOTO 10
```

(Bemærk, at vi ikke behøver en linje i begyndelsen af programmet, som giver a "startværdien" nul. Den har den i forvejen). Den sidste version har yderligere den fordel, at der kan pyntes lidt på den:

```
10 a=a+1
20 PRINT a;"gange 7 er";a*7
30 GOTO 10
```

Når skærmen er fuld, kan vi taste ESC og fortsætte med f.eks. ENTER. Når vi ikke vil gå højere, bringer to gange ESC os ud af programmet. Lidt mere kompliceret er

```
10 a=1985
20 b=100
30 PRINT a ; b
40 a=a+1
50 b=b*1.09
60 GOTO 30
```

Situationen, vi har skrevet om til et program, er den, at vi i 1985 sætter 100 kroner ind til 9% rente. Vi vil nu gerne have oplyst indestående de følgende år.

Ja, der er to ting, maskinen skal holde rede på, den ene, hvad år det er, den anden, hvor mange penge vi har. Hvad med rentefoden? Den er konstant, så det er tilstrækkeligt, hvis den bare står et eller andet sted i programmet (i linje 50).

År og indestående varierer imidlertid, de antager hele tiden nye værdier, og derfor må vi have variabler til at tage sig af dem, nærmere betegnet to af dem, som vi kan kalde a og b. Vi kunne også kalde dem "anno" og "penge", men a og b er billigere i hukommelse og giver navnlig kortere og derfor mere overskuelige programmer.

Først må vi skrive, hvad a og b er til en begyndelse, deres "startværdier". Det kalder vi "startstilling". Dette programs startstilling udgøres af linjerne 10 og 20.

Nu kan vi begynde at regne. Vi starter med at skrive årstal og beløb, som de er, før vi begynder at regne på dem.

I linje 40 lader vi så et år gå (a bliver 1 større). Hvad sker der nu med b? Jo, der bliver lagt 9% til beløbet, vi lægger beløbet * 0.09 til eller ganger med 1.09, hvilket giver det samme resultat.

Mere skal der ikke gøres, det skal bare gøres igen og igen. Derfor sætter vi et GOTO i linje 60.

Læg mærke til, at det sender os tilbage til 30, hvor det nye beløb skrives. Hvis vi var blevet sendt længere tilbage, ville beløbet igen blive 100 kroner og eventuelt året 1985,

og vi ville aldrig komme videre. Lad maskinen skrive skærmen fuld og tast ESC to gange.

Vi vil nu se, at vi i 1986 har 109 kroner og i 1987 118 kroner og 81 øre. Men hvad med 1988?

129.5029? Tak, to decimaler er nok. Men det kan maskinen også klare, nemlig med USING. Skriv

```
PRINT USING ".##"; 123.4567
```

Maskinen skriver

```
%123.46
```

Læg mærke til, at maskinen ikke blot fjerner de overflødige decimaler, den runder dem også op. Prøv tilsvarende USING ".#" og ".###". Vi kan nu lave vor linje 30 om til

```
30 PRINT a; USING ".##"; b
```

Stop maskinen ved år 2100. Det var et ordentligt beløb, og det kan gøres lettere læseligt, hvis vi endnu en gang retter 30, nemlig til

```
30 PRINT a; USING "#,###.##"; b
```

Læg mærke til % foran tallene, når de overskrider formatet "#,###.##". Det angiver, at de ikke længere står komma under komma. For at gøre det skal maskinen nemlig vide, hvor høje tal den kan komme ud for.

#,###.## betyder således, at det højeste er 9,999.99, og det er jo ikke rigtigt. Det kunne vi så ændre ved at definere formatet som
"###,###,###.##".

Som du vil forstå, skriver man "på engelsk" 9,999.99,

hvor vi ville skrive 9.999,99. Kør igen og stop denne gang ved 2173. Hvad søren er nu

1.08688001E + 09

Ja, for at undgå at skrive en skrækkelig masse nuller, anvender maskinen en lidt anden skrivemåde. E + 09 betyder, at kommaet (som altså på maskinen er et punktum), skal flyttes 9 pladser til højre. Det rigtige tal er altså

1,086,880,010.00

eller noget over en milliard. Kender du historien om skakspillets opfinder, der af sin konge forlangte 1 korn for det første felt, 2 for det andet, 4 for det tredje, 8 for det fjerde, osv.? Skal vi prøve at se, hvor meget korn han egentlig fik ud af det? Prøv

```
10 a=1
20 b=1
30 PRINT "For felt nr."; a ;"betales" ; b
40 a=a+1
50 b=b*2
60 GOTO 30
```

Kør programmet, og vi får

```
For felt nr. 1 betales 1
For felt nr. 2 betales 2
For felt nr. 3 betales 4
For felt nr. 4 betales 8
................................
For felt nr. 64 betales 9.22337E+18
```

eller 9,223,370,000,000,000,000 korn eller knap ti milliarder milliarder. Det er nok overordentlig tvivlsomt, om kongen vil være i stand til at betale. Nu kunne han jo have stillet det modforslag, at opfinderen skulle have et helt vognlæs korn for det første felt, men kun et halvt vognlæs for det andet, en fjerdedel for det tredje, osv. For at få dette nye program behøver vi såmænd blot udskifte 50 med

```
50 b=b / 2
```

der giver

```
For felt nr. 1 betales 1
For felt nr. 2 betales 0.5
For felt nr. 3 betales 0.25
For felt nr. 4 betales 0.125
.......................................
For felt nr. 64 betales 1.08420E-19
```

Som du måske har regnet ud, skal kommaet nu flyttes 19 pladser til *venstre*, og vi får 0.0000000000000000010842 vognlæs for det sidste felt, mindre end en milliontedel af en milliontedel af en milliontedel læs. Men hvad skal nu kongen af med i alt? Prøv

```
10 a=1
20 b=1
30 c=c+b
40 PRINT "Op til felt nr."; a ; "betales" ; c
50 a=a+1
60 b=b / 2
70 GOTO 30
```

Den ny variabel c lægger hele tiden beløbet for det næste felt til, og det, der printes ud i 40, er følgelig den *samlede* sum for de udregnede felter. Vi får da

Op til felt nr. 1 betales 1
Op til felt nr. 2 betales 1.5
Op til felt nr. 3 betales 1.75
Op til felt nr. 4 betales 1.875

Op til felt nr. 26 betales 1.99999997 vognlæs, op til nr. 27 1.99999999, og fra 29 rundes der simpelt hen op til 2. Kongen kunne aldrig komme af med mere end højst to vognlæs! Det første felt på brættet kostede ham til gengæld lige så meget som alle de andre til sammen. Lad os vende tilbage til vores rentesregning:

```
10 a = 1985
20 b = 100
30 PRINT a ; b
40 a = a + 1
50 b = b * 1.09
60 GOTO 30
```

Hvad nu, hvis vi ville have sat 200 kroner ind til 7% rente i 1990? Ja, så skal vi til at skrive hele programmet om, og det er selvfølgelig ikke så praktisk. Det program, vi har skrevet, kan nemlig kun bruges på 100 kroner, 9% og 1985. Men prøv så

```
10 INPUT a
20 PRINT a
30 PRINT a * 2
40 INPUT a$
50 PRINT a$
```

Kør programmet. Maskinen skriver blot et spørgsmålstegn, og der kommer ikke noget Ready. Den venter.

Hvad venter den på? At du skal fortælle den, hvad a skal være.

Skriv 27 og tast ENTER. At maskinen har forstået, at a skal være 27, kan du se ved, at den skriver tallet og derefter som forlangt i linje 30 tallet ganget med 2. Så kommer der et nyt INPUT, og maskinen venter igen.

Skriv amstrad (ENTER), og maskinen gentager det som forlangt i 50. Læg mærke til, at du ikke behøver at skrive gåseøjne. Kan du se fidusen? Tag blot de første to linjer

```
10 INPUT a
20 PRINT a*2
```

Dette program vil fordoble ethvert tal, du taster ind. Du behøver altså ikke rette i programmet, fordi det nu f.eks. er 234, du vil have ganget med to. Endnu mere praktisk er

```
10 INPUT a
20 INPUT b
30 PRINT a ; "gange" ; b ; "er" ; a*b
```

Et andet eksempel kunne være

```
10 INPUT a$
20 INPUT a
30 PRINT "Du hedder "; a$ ;" og er" ; a
```

Dette eksempel virker selvfølgelig kun, hvis du svarer på det første spørgsmålstegn med dit navn og på det andet med din alder, og hvor kan du nu vide, at du skal det? Du kunne indføje linjerne

```
 5 PRINT "Navn?"
15 PRINT "Alder?"
```

men disse spørgsmål kan også inkluderes i selve INPUT-linjen:

```
10 INPUT "Navn"; a$
20 INPUT "Alder"; a
30 PRINT "Du hedder "; a$;" og er"; a
```

Prøv at svare omvendt på spørgsmålene. I 10 går det meget godt (måske hedder du virkelig 98), men i 20 går det galt. Maskinen siger

```
?Redo from start
Alder?
```

om igen, soldat. Maskinen bad om et tal, og du har givet den en streng. Indtast nu blot et tal, og den er glad igen. I øvrigt kan en INPUT-linje klare mere end et enkelt INPUT. Vi kunne sagtens undvære 20 ved at lave 10 om til

```
10 INPUT "Navn, alder"; a$, a
```

Prøv også

```
10 INPUT "5 tal"; a, b, c, d, e
20 PRINT a; b; c; d; e
```

Svar f.eks.

```
23, 45, 67, 89, 12
```

altså, imellem de enkelte INPUT. Du kunne lave 20 om til

```
20 PRINT a+b+c+d+e
```

eller bedre

```
10 INPUT a
20 b=b+a
30 PRINT "Sum";b
40 GOTO 10
```

Kan du se, hvad programmet gør? Ja, det lægger tal sammen ligesom et kasseapparat. Hvorfor ville det ikke have nyttet med RUN i 40? Vi kan nu meget praktisk lave vores renteprogram om til

```
10 INPUT "ANNO, BELØB, RENTEFOD I %";a,b,c
20 PRINT a; USING "#,###.##";b
30 a=a+1
40 b=b+b*c/100
50 GOTO 20
```

Det begynder at ligne programmering, dette her. Ja, tænk, hvis vi havde sat 1000 kroner ind i 1950 til 5%, så ville vi i dag have ... 5516 kroner og 2 øre. Indsæt nu

```
25 INPUT z
```

Maskinen skriver

```
1950 1,000.00
?
```

Den venter jo på værdien af z. Men hvad skal vi bruge z til? Intet som helst andet end at bremse det løbske program. Tast blot ENTER, og maskinen skriver

```
1951  1,050.00
?
```

Hver gang du taster ENTER, får du et nyt indestående. Telegrafer hurtigt med ENTER, til du har en skærmfuld, som du så i ro og mag kan betragte. Her er et (meget) lille spil:

```
 10 INPUT "Navne"; a$, b$
 20 PRINT a$;
 30 INPUT " skriver"; c$
 40 d$=d$+c$
 50 PRINT "Ordet er nu "; d$
 60 PRINT b$;
 70 INPUT " skriver"; c$
 80 d$=d$+c$
 90 PRINT "Ordet er nu "; d$
100 GOTO 20
```

Læg mærke til, at mange linjer er ens. Du kan altså spare dig selv for en del arbejde ved, når du f.eks. har skrevet 30, at skrive EDIT, rette linjenummeret til 70 og taste ENTER.

Ikke sandt? Sådan "fordobler" du linjer.

Du kan også blive af med dem, simpelt hen ved at skrive linjenummeret fulgt af ENTER. Det opfatter maskinen som en "tom" linje. Endelig kan du fjerne større dele af programmet.

```
DELETE 20-80
```

ville skaffe os af med det meste af vores program, så lad være med at skrive det, medmindre du er parat til at skrive programmet en gang til. Men lad os nu prøve vores spil. Vi skal bruge to deltagere, og maskinen starter med at bede om deres navne:

Navne ?

Vi skriver

Peter , Povl

fulgt af ENTER, og spillet begynder. Peter skal skrive et bogstav, og han vælger i. Maskinen svarer:

Ordet er nu i
Povl skriver ?

Det gælder nu for Povl om at skrive et bogstav, så der dannes et ord. Han klarer sig med l, Peter svarer med t, og spillet fortsætter

Navne ? Peter , Povl
Peter skriver ? i
Ordet er nu i
Povl skriver ? l
Ordet er nu il
Peter skriver ? t
Ordet er nu ilt
Povl skriver ? e
Ordet er nu ilte
Peter skriver ? r
Ordet er nu ilter
Povl skriver ? t
Ordet er nu iltert
Peter skriver ?

Av, nu giver Peter op. Første runde til Povl, ESC og RUN, og næste omgang er begyndt.

Du husker måske fra din skoletid en vis våd vittighed om dyr og tal? Det er en skrækkelig dårlig vittighed (hvis man

er mere end syv år gammel), men som programeksempel er den udmærket.

```
10 INPUT "Hvilket dyr kan du bedst lide"; a$
20 NPUT "Hvad er dit lykketal"; a
30 PRINT "Har du nogen sinde set en "; a$;" med
   "; a; "ben ?"
```

Vi kan selvfølgelig også nu få enhver tabel, vi måtte forlange:

```
10 INPUT "Tabel"; a
20 b=b+1
30 PRINT b; "gange"; a; "er"; a*b
40 INPUT c
50 GOTO 20
```

Eller hvad med

```
10 INPUT "Tabel"; a
20 b=b+1
30 PRINT "Hvad er"; b; "gange"; a;
40 INPUT c
50 PRINT b; "gange"; a; "er"; a*b
60 GOTO 20
```

Maskinen spørger

```
Tabel ?
```

og vi kan svare f.eks. 7. Maskinen reagerer med

```
Hvad er 1 gange 7 ?
```

Vi svarer og checker med det rigtige svar. Det kan komme til at se sådan ud på skærmen:

```
Tabel ? 7
Hvad er 1 gange 7 ? 7
 1 gange 7 er 7
Hvad er 2 gange 7 ? 14
 2 gange 7 er 14
Hvad er 3 gange 7 ? 22
 3 gange 7 er 21
Hvad er 4 gange 7 ? 28
 4 gange 7 er 28
```

3 gange 7 kunne vi ikke (det *er* også svært), og det kunne vi konstatere, eftersom maskinen hele tiden skriver det rigtige, lige så snart vi har svaret, hvad enten vi har svaret rigtigt eller ej. Maskinen aner det ikke, den bruger overhovedet ikke vores c til noget som helst. Kunne det nu ikke være rart, hvis den rent faktisk kunne konstatere, om der var svaret rigtigt? Man kunne måske forestille sig en linje i stil med

```
Hvis c=a*b så PRINT "Rigtigt!"
```

Hvis vi oversætter til engelsk, har vi faktisk linjen, som vi kan indsætte:

```
45 IF c=a*b THEN PRINT "Rigtigt!"
```

Men hvis vi har svaret rigtigt, hvad skal vi så med den oplysning om det rigtige svar, der følger efter? Det, vi skal bruge, er noget i retning af

```
IF c=a*b THEN PRINT "Rigtigt!" og derefter
GOTO 20, videre med den næste opgave!
```

Men kan vi få to "følger" af, at den samme betingelse er opfyldt? Det kan vi godt, hvis vi sætter kolon imellem. Skriv

```
45 IF c=a*b THEN PRINT "Rigtigt!": GOTO 20
```

Fikst, ikke? Vi kunne så eventuelt yderligere rette 50 til

```
50 PRINT "Nej,"; a*b
```

Hvor længe bliver det cirkus så ved? Tja, vi kan jo ikke egentlig stoppe det, kun "snyde" ved at bryde ud af programmet med ESC, når vi ikke gider mere. Eller kan vi? Prøv

```
25 IF b=11 THEN END
```

Hvad står der her? END betyder, at programmet skal stoppe.

Hvornår? Når b er blevet 11, hvilket betyder, at det sidste spørgsmål bliver 10 gange 7.

Se, den fidus kunne vi jo godt have brugt noget før. F.eks. kunne vi have fået vores kornprogram til at stoppe, når felt nr. 64 var nået. Prøv så

```
10 INPUT "Tabel"; a
20 IF b=10 THEN 90
30 b=b+1
40 PRINT "Hvad er"; b; "gange"; a;
50 INPUT c
60 IF c=a*b THEN PRINT "Rigtigt!": d=d+1:
   GOTO 20
70 PRINT "Nej,"; a*b
80 GOTO 20
90 PRINT "Du havde"; d; "rigtige!"
```

Vi kan analysere programmet som følger:

10 Maskinen spørger, hvilken tabel vi vil høres i, og får et svar.
20 Hvis vi er nået op til (og er færdige med) 10 gange, går maskinen ud af sit kredsløb 20-80 (sådan et kredsløb kaldes, som vi ved, for en løkke) og ned i 90. Læg mærke til, at der er en "indre løkke" 20-60 for det tilfældes skyld, at vi svarer rigtigt. Hvis betingelsen *ikke* er opfyldt, går maskinen som altid videre til næste linje.
30 Maskinen tæller 1. Læg mærke til, at b er 0 til at begynde med. Ved første gennemløb bliver den altså 1.
40 Maskinen stiller sit spørgsmål.
50 Maskinen får svar.
60 Hvis svaret er rigtigt, sker der 3 ting:
   1. Maskinen skriver "Rigtigt!"
   2. d, en "tæller", der tæller rigtige besvarelser (vi skal bruge den i 90), tæller 1.
   3. Maskinen går tilbage til 20 og forbereder en ny omgang.
   Hvis svaret *ikke* er rigtigt, falder vi ned i 70.

70 Maskinen skriver "Nej" og det rigtige resultat.
80 Tilbage igen.
90 Maskinen skriver, hvor mange gange vi svarede rigtigt.

Som sagt går maskinen videre til næste linje, hvis betingelsen ikke er opfyldt. Men vi kan også fortælle den, hvad den skal gøre. Vi bruger så en form af typen

HVIS ... SÅ ... ELLERS

eller på engelsk

IF ... THEN ... ELSE

Vores program kunne da blive til

```
10 INPUT "Tabel"; a
20 IF b=10 THEN 80
30 b=b+1
40 PRINT "Hvad er"; b;"gange"; a;
50 INPUT c
60 IF c=a*b then PRINT "Rigtigt!": d=d+1
   ELSE PRINT "Nej, "; a*b
70 GOTO 20
80 PRINT "Du havde"; d;"rigtige!"
```

Læg mærke til, at vi ikke i 20 behøver at skrive THEN GOTO 80, THEN 80 er nok. Med ELSE sparede vi en linje. Men vi kan spare mere endnu. Prøv

```
10 INPUT "Tabel"; a
20 IF b=10 THEN 50
30 b=b+1: PRINT "Hvad er"; b; "gange"; a;:
   INPUT c: IF c=a*b THEN PRINT "Rigtigt!":
   d=d+1 ELSE PRINT "Nej,"; a*b
40 GOTO 20
50 PRINT "Du havde"; d;"rigtige!"
```

Vi kan sådan set slå lige så mange linjer sammen med kolon, vi vil. Der er kun to begrænsninger.

For det første kan vi selvfølgelig ikke hægte flere linjer på et THEN, så bliver de nemlig følger af betingelsen efter IF. For det andet må en linje ikke overstige 255 karakterer. Prøv

```
10 INPUT "tim, min, sek"; t, m, s
20 CLS: PRINT t; m; s: s=s+1: IF s=60 THEN
   s=0: m=m+1
30 IF m=60 THEN m=0: t=t+1
```

```
40 IF t=24 THEN t=0
50 p=p+1 : IF p=209 THEN p=0 : GOTO 20
60 GOTO 50
```

Kan du se, hvordan det virker? 50-60 er en løkke, maskinen kører rundt i, til den har talt til 209 (så er der nemlig gået cirka et sekund).

Så går den op i 20, og hvad gør den der? CLS er en ny ordre, der betyder "slet skærmen".

Så skriver den tidspunktet i timer, minutter og sekunder, og s bliver en større. Hvis s er 60, bliver den 0 igen, og m en større (der er gået et minut), og tilsvarende skiftes time ved m = 60.

Når t er 24, bliver den 0 igen, og uret begynder forfra.

Prøv at køre det. Maskinen siger

```
tim , min , sek ?
```

Vi indtaster nu det aktuelle klokkeslæt, f.eks. 10 , 11 , 30. I det øjeblik, vi taster ENTER, går uret i gang.
Prøv

```
10 PRINT "AMSTRAD" : a=a+1 : IF a=10 THEN
   END
20 GOTO 10
```

Maskinen skriver AMSTRAD, tæller 1 på a, skriver det igen, tæller igen, og så videre, indtil a er 10, hvor den stopper. Maskinen skriver med andre ord AMSTRAD 10 gange.
= er en relation. Det er imidlertid ikke den eneste, der er seks af dem ialt, nemlig

=	lig med
<	mindre end
>	større end

< =	mindre end eller lig med
> =	større end eller lig med
< >	forskellig fra (mindre eller større end)

Vi kan nu meget praktisk skrive vores program om til en enkelt linje, nemlig

```
10 PRINT "AMSTRAD" : a = a + 1 : IF a < 10 THEN 10
```

Vi har i de to forløbne kapitler undersøgt to meget vigtige begreber inden for programmering, *variabel* og *program*. Et tredje og lige så vigtigt skal vi hellige os i det næste kapitel, nemlig *funktion*.

Vi er også blevet bekendt med tastaturet, og en del ordrer kender vi nu ligeledes til. Kan du huske dem?

PRINT	bruger vi, når vi skal have noget skrevet på skærmen
CLEAR	sletter alle variabler
RUN	kører vores program
LIST	giver os programlisten
NEW	sletter programmet
CONT	fortsætter programkørslen, når vi har standset den med ESC (2 gange ESC – hvis vi kun har tastet det en enkelt gang, kan vi fortsætte med enhver tast)
EDIT	sætter os i stand til at redigere programlinjer
GOTO	bringer maskinen et andet sted hen i programmet og får den til at fortsætte der
USING	påvirker den måde, tallene bliver skrevet ud på med PRINT
INPUT	tillader os at indtaste tal og strenge i programmet

DELETE sletter en udvalgt del af programmet
IF THEN ELSE svarer cirka til hverdagssprogets hvis så ellers
CLS sletter skærmen

Det er med andre ord på tide, at maskinen viser os, hvor smart den er. Hvad med

```
10 PRINT "JEG FINDER DIT TAL(< =100), HVIS
   DU GØR, SOM JEG SIGER": INPUT
   "DIVIDER DET MED 3! REST"; a : INPUT
   "DIVIDER DET MED 5! REST"; b : INPUT
   "DIVIDER DET MED 7! REST"; c : d=a*70+
   b*21+c*15
20 IF d> 105 THEN d=d-105 : GOTO 20
30 PRINT "DIT TAL ER"; d
```

Prøv at køre det. Maskinen skriver

```
JEG FINDER DIT TAL (< =100), HVIS DU GØR,
SOM JEG SIGER
DIVIDER DET MED 3! REST?
```

Lad os sige, vi har tænkt på 37. 3 op i 37 er 12, men det afgørende er, at vi får 1 til rest. Vi svarer altså 1, og programmet fortsætter

```
DIVIDER DET MED 3! REST? 1
DIVIDER DET MED 5! REST? 2
DIVIDER DET MED 7! REST? 2
DIT TAL ER 37
```

Virker garanteret hver gang! Eller prøv

```
10 a=25 : PRINT "DER LIGGER 25 GLASKUGLER
   I EN TALLERKEN. VI SKIFTES TIL AT TAGE
   ET ANTAL, MINDST 1 OG HØJST 3. DEN,
   DER TAGER DEN SIDSTE, TABER."
20 PRINT "DER ER"; a; "I TALLERKEN": IF a=1
   THEN PRINT "DU TABER!": END
30 INPUT "DU TAGER"; b: IF b<1 THEN 30
40 IF b>3 THEN 30
50 c=4-b: PRINT "JEG TAGER"; c: a=a-4:
   GOTO 20
```

Vi prøver

```
DER LIGGER 25 GLASKUGLER I EN TALLERKEN.
VI SKIFTES TIL AT TAGE ET ANTAL, MINDST 1
OG HØJST 3. DEN, DER TAGER DEN SIDSTE,
TABER.
DER ER 25 I TALLERKENEN
DU TAGER? 1
JEG TAGER 3
DER ER 21 I TALLERKENEN
DU TAGER? 2
JEG TAGER 2
DER ER 17 I TALLERKENEN
DU TAGER? 3
JEG TAGER 1
DER ER 13 I TALLERKENEN
DU TAGER? 1
JEG TAGER 3
DER ER 9 I TALLERKENEN
DU TAGER? 2
JEG TAGER 2
DER ER 5 I TALLERKENEN
DU TAGER? 3
```

```
JEG TAGER 1
DER ER 1 I TALLERKENEN
DU TABER !
```

Kan du vinde over maskinen? Lidt fornøjelse kan vi da allerede have ud af den. Når vi nu får funktionerne med også, begynder der for alvor at ske noget.

3. FUNKTIONER OG LOGISKE OPERATIONER

SQR – INT – CINT – ROUND – RND – RANDOMIZE – MODE – AND – OR – NOT – FOR – NEXT – STEP – LOCATE

I dette kapitel skal vi se på funktioner. Skriv

```
? SQR(49)
```

Maskinen svarer 7. SQR står for SQuare Root eller kvadratrod. Hvad er en kvadratrod?

Hvis vi skal købe et gulvtæppe til et firkantet værelse, der er lige så langt, som det er bredt, altså *kvadratisk*, kan vi opmåle en af væggene (siden i kvadratet) og f.eks. få den til 7 meter. Gulvtæpper køber man imidlertid i kvadratmeter, det vil sige kvadrater med siden 1 meter.

Sådan nogle små kvadrater kan vi få 49 af i værelset. Vi har sagt 7 * 7 = 49. 49 er *kvadratet på* 7.

Er vi nu imidlertid i den modsatte situation og får opgivet flademålet 49 og vil finde siden i kvadratet, må vi også gå den modsatte vej og finde "roden" til de 49. Kvadratroden af 49 er 7, fordi 7 *ganget med sig selv* er 49.

Det var jo ikke så svært. Men sæt nu, gulvet i vores (stadig væk kvadratiske) værelse er 50 kvadratmeter. Så må vi have maskinen til hjælp og spørge

```
? SQR(50)
```

Væggen er altså lidt over 7 meter og 7 centimeter. Det modsatte af rod er forresten potens.

7 ganget med sig selv kan vi også kalde 7 i *anden potens*. 7 * 7 * 7 bliver 7 i tredje potens, og så fremdeles.

Potenstegnet er pilen under pundtegnet på tastaturet.

? 7↑2 , 7↑3

Kvadratrod er en *funktion*, og hvad er en funktion så? Ikke særlig matematisk kan vi udtrykke funktionen som noget, der gør noget ved et tal, så det bliver til et andet tal. 49 blev til 7, ikke? Prøv

? INT(7.07106781)

INT betyder INTeger eller "helt tal", og funktionen laver et tal om til et helt tal. Læg mærke til, at maskinen blot smider decimalerne væk. CINT runder op. Prøv

? INT(0.5) , CINT(1.5)

ROUND er endnu en version. Skriv

? ROUND(1.23456789 , 3)

Maskinen skriver 1.235, for 3-tallet betyder, at du vil have 3 decimaler. Prøv nu

? ROUND(123456789 , – 3)

Nu skriver den 123457000, for –3 betyder, at du kun vil have de hele tusinder. –6 giver dig kun millioner (resten er jo småpenge).

En lidt speciel og meget vigtig funktion er RND, der står for RaNDom = tilfældig. RND gør ikke noget ved noget tal (den har intet *argument*), den laver bare et tilfældigt tal. Prøv

? RND

Maskinen skriver 0.271940658. Hvordan kan jeg nu vide det, hvis det er tilfældigt? Ja, sådan rigtig tilfældigt er det jo ikke, det *virker* bare tilfældigt, fordi vi ikke kan gennemskue den måde, maskinen regner sig frem til det på, og 0.271940658 er det første tal, maskinen kommer i tanker om, når vi tænder for maskinen. Prøv

```
? RND , RND , RND
```

Så fik vi tre nye. Vil vi have maskinen til at starte med et andet tal, kan vi bruge RANDOMIZE.

```
RANDOMIZE 27
```

vil f.eks. altid få maskinen til at starte med 0.646940658. Læg mærke til, at tallet altid ligger mellem 0 og 1. Men det kan vi jo nemt rette. Prøv

```
? RND*100
```

Nu får vi et tal mellem 0 og 100. Decimalerne kan vi komme af med med INT. Prøv

```
? INT(RND*100)
```

Vi får et helt tal, fra 0 op til 99. Vi kan ikke få 100, for RND bliver aldrig helt 1, og INT smider jo decimalerne væk. Kunne vi eventuelt lave os en terning?

```
? INT(RND*6)
```

ville ikke gøre det, for det giver mulighederne 0-5. Løsningen må altså blive

```
? INT(RND*6)+1
```

Vi er nu ved at kunne lave nogle rigtige spil. Prøv f.eks.

```
10 a = 1000 : b = 10000
20 IF a = 0 THEN PRINT "Du er fallit" : END
30 IF b< =0 THEN PRINT "Banken er fallit" : END
40 PRINT : PRINT "Kapital :" ; a
50 INPUT "Tal (0–9)" ; c : INPUT "Indsats" ; d : IF
   d>a THEN 50
60 a = a – d : b = b + d : e = INT(RND * 10) : PRINT e ;
   "kom ud" : IF e = c THEN PRINT "Du vandt" :
   a = a + d * 10 : b = b – d * 10 : GOTO 20
70 PRINT "Du tabte" : GOTO 20
```

Lad os engang prøve:

```
Kapital: 1000
Tal (0–9) ?
```

Vi har altså 1000 kroner at spille for, og vi kan holde på et tal fra 0 til 9. Lad os prøve med 7.

```
Kapital : 1000
Tal (0–9) ? 7
Indsats ? 100
2 kom ud
Du tabte

Kapital: 900
Tal (0–9) ? 7
Indsats ? 50
5 kom ud
Du tabte

Kapital : 850
Tal (0–9) ?
```

Lad os prøve 1 i stedet for:

Kapital : 850
Tal (0–9) ? 1
Indsats? 200
0 kom ud
Du tabte

Kapital : 650
Tal (0–9) ? 1
Indsats ? 200
1 kom ud
Du vandt

Kapital : 2450
Tal (0–9) ?

og så videre. Men hvordan virker det? Vi kunne analysere det cirka som følger:

10 Sæt egen kapital og bankens kapital.
20 Hvis din kapital bliver 0, er du fallit, og spillet ender.
30 Hvis bankens kapital slipper op, er banken fallit, og spillet ender.
40 Skriv, hvor mange penge du har.
50 Tag imod bestilling, dels tallet, du spiller på, dels indsatsens størrelse. Hvis indsatsen er mere, end du har i banken, om igen.
60 Din indsats trækkes fra din kapital og lægges til bankens ditto. Lykkehjulet snurrer, og resultatet skrives op. Hvis dit og hjulets tal stemmer over ens, vandt du, og du skal have din indsats tidobbelt igen, og denne sum skal trækkes fra bankens kapital. Forfra igen.
70 Hvis ikke, tabte du. Forfra igen.

70 kunne selvfølgelig være blevet et ELSE på 60, det er en smagssag. Måske synes du, det hele står så underligt og flagrer ude til venstre. Prøv at indsætte

```
5 MODE 0
```

MODE 2 ville have givet mindre skrift, så du kunne have mere på skærmen. Du kan også prøve

```
 10 PRINT "VI LEGER FIND TALLET. DIT TAL
    LIGGERmmmmmIMELLEM 1 OG";: INPUT
    a:b=a:PRINT "OKAY,";a:PRINT "HVIS DIT
    TAL ER STØRRE, SKAL DU SKRIVEmm s;
    HVIS DET ER MINDRE, m; HVIS DET
    ERmmm RIGTIGT, r":c=1:d=INT((b-c)/2)+1
 20 e=e+1: PRINT e;". FORSØG:";d:IF b=c
    THEN 80
 30 INPUT a$: IF a$="r" THEN 80
 40 IF a$="s" THEN c=d+1:d=d+INT((b-d)/2)
 50 IF a$="m" THEN b=d-1:d=c+INT((d-c)/2)
 60 IF b-d=1 THEN d=d+1
 70 GOTO 20
 80 PRINT "JEG FANDT";d;"I";e;"FORSØG":
    PRINT "KAN DU GØRE DET BEDRE? MIT
    TAL ERmmmmmmmmLIGELEDES MELLEM 1
    OG";a:PRINT "OPGIVER DU, KAN DU
    TASTE 0":b=INT(RND*a)+1
 90 INPUT a:IF a=0 THEN 180
100 f=f+1:PRINT f;". FORSØG:";a:IF a=b
    THEN 140
110 IF b>a THEN PRINT "s"
120 IF b<a THEN PRINT "m"
130 GOTO 90
140 PRINT "DU FANDT MIT TAL I";f;
```

```
     "FORSØG": PRINT "JEG FANDT DIT I"; e:
     IF e<f THEN PRINT "JEG VANDT"
150 IF e>f THEN PRINT "DU VANDT"
160 IF e=f THEN PRINT "UAFGJORT"
170 END
180 PRINT "OKAY, DET VAR"; b: PRINT "JEG
     VANDT"
```

Der er en hel del m'er i linje 10 og 80. Når der er mere end et, kan du være temmelig sikker på, at det betyder mellemrum. Du skal altså f.eks. taste 7 mellemrum mellem ER og LIGELEDES i linje 80 (for at få det til at stå pænt på skærmen). Lad os prøve at spille det:

```
VI LEGER FIND TALLET. DIT TAL LIGGER
IMELLEM 1 OG ? 1000
OKAY, 1000
HVIS DIT TAL ER STØRRE, SKAL DU SKRIVE
s; HVIS DET ER MINDRE, m; HVIS DET ER
RIGTIGT, r
 1 . FORSØG: 500
? s
 2 . FORSØG: 750
? m
 3 . FORSØG: 625
? s
 4 . FORSØG: 687
? m
 5 . FORSØG: 656
? s
 6 . FORSØG: 671
? m
 7 . FORSØG: 664
? s
```

```
  8 . FORSØG : 667
? m
  9 . FORSØG : 666
? r
JEG FANDT 666 I 9 FORSØG
KAN DU GØRE DET BEDRE ? MIT TAL ER
LIGELEDES MELLEM 1 OG 1000
OPGIVER DU , KAN DU TASTE 0
?
```

og så er det vores tur ...

AMSTRAD har mange andre funktioner, men dem skal vi kunne programmere lidt mere for at få rigtig glæde af, så dem gemmer vi lidt.

Er potens en funktion? Ja, den laver da 7 om til 49, ikke? Egentlig ikke. 7↑2 er jo kun en slags "forkortelse" for 7 * 7, og * kaldes ligesom +, – og / for *operationer*. Vi skal nu se på en særlig slags, de *logiske operationer*. Prøv

```
10 INPUT "STØRSTE PLANET"; a$: IF a$="jupiter"
   THEN END
20 RUN
```

Programmet stiller opgaven, til det rigtige svar indtastes. Prøv nu

```
10 INPUT "MARS' DRABANTER"; a$, b$: IF a$=
   "phobos" AND b$="deimos" THEN END
20 RUN
```

Nu standser programmet kun, hvis *begge* svar er rigtige. Men sæt nu, eleven svarer i omvendt rækkefølge? Så må vi have

```
10 INPUT "MARS' DRABANTER";a$,b$: IF a$=
   "phobos" AND b$="deimos" OR a$="deimos"
   AND b$="phobos" THEN END
20 RUN
```

OR betyder eller. Hvis blot en af betingelserne er opfyldt, standser programmet, altså enten

1. a$="phobos" og b$="deimos"

eller

2. a$="deimos" og b$="phobos"

Men hvordan kan maskinen nu vide, at vi ikke mener, at programmet skal stoppe, hvis det er opfyldt, at

1. a$="phobos"

og

2. b$="deimos" eller a$="deimos"

og

3. b$="phobos"

Dette ville jo aldrig ske, for hvis a$ og b$ begge er "phobos", kan ingen af dem være "deimos". Svaret er igen "prioritering". Maskinen tager simpelt hen alle AND, før den begynder med OR.

Der er imidlertid en logisk operation (eller rettere funktion), den udfører før både AND og OR, og det er NOT: ikke. Lad os igen se på vores første eksempel. Faktisk kunne vi have nøjedes med en enkelt linje:

```
10 INPUT "STØRSTE PLANET"; a$: IF NOT a$=
   "jupiter" THEN RUN
```

Nu siger programmet i stedet for, at maskinen skal spørge igen, hvis svaret *ikke* er rigtigt, og det kommer jo faktisk ud på et. Kunne vi gøre det samme med det næste eksempel:

```
10 INPUT "MARS' DRABANTER"; a$, b$: IF NOT
   (a$="phobos" AND b$="deimos") THEN RUN
```

Hvad skal vi med parentesen? Ja, hvad ville

```
10 INPUT "MARS' DRABANTER"; a$, b$: IF NOT
   a$="phobos" AND b$="deimos" THEN RUN
```

betyde? Nemlig: Spørg igen, hvis det er opfyldt, at

1. a$ ikke er "phobos"

og

2. b$ er "deimos"

Maskinen tager NOT før AND, ikke? Måske synes du, linjen ser lidt mystisk ud. Kunne vi ikke omskrive den på en måde, så vi kan undvære parentesen? Hvad med

```
10 INPUT "MARS' DRABANTER"; a$, b$: IF NOT
   a$="phobos" AND NOT b$="deimos" THEN
   RUN
```

Det ser da plausibelt ud, ikke? Men pas nu på: Det betyder jo, at *både* phobos *og* deimos skal være forkerte, hvis maskinen skal gå tilbage. Og da vi jo må forlange begge rigtige,

for at maskinen skal stoppe, skal også den ene forkerte være nok til at sende maskinen tilbage.

```
10 INPUT "MARS' DRABANTER"; a$, b$: IF NOT
   (a$="phobos" AND b$="deimos") THEN RUN
```

må altså oversættes ved

```
10 INPUT "MARS' DRABANTER"; a$, b$: IF NOT
   a$="phobos" OR NOT b$="deimos" THEN
   RUN
```

Hvad nu, hvis linjen havde heddet

```
10 INPUT "MARS' DRABANTER"; a$, b$: IF NOT
   (a$="phobos" OR b$="deimos") THEN RUN
```

Hvornår ville maskinen så være gået tilbage? Når det ikke var tilfældet, at (a$="phobos" OR b$="deimos"). Med andre ord, når *ingen af delene* var tilfældet. Dette udtrykker altså i virkeligheden et "hverken-eller" og kan oversættes

```
10 INPUT "MARS' DRABANTER"; a$, b$: IF NOT
   a$="phobos" AND NOT b$="deimos" THEN
   RUN
```

Vi kan også udtrykke os lidt mere generelt, idet vi lader bogstaverne P og Q (en slags variabler) stå for hele relationer. Så kan vi skrive

NOT (P AND Q) er det samme som NOT P OR NOT Q
NOT (P OR Q) er det samme som NOT P AND NOT Q

Husk endelig på, at

```
NOT a=b
```

er præcis det samme som

```
a< >b
```

På samme måde er a<b det modsatte af a> =b, og a>b det modsatte af a< =b.

Vi har i dette kapitel set på funktionsbegrebet og de logiske operationer. Måske synes du ikke umiddelbart, du kan bruge nogen af delene til ret meget.

Tag nu f.eks. INT. Hvad i alverden skal man bruge den til? Lad os se, om vi kan finde på noget.

Jeg bliver ofte spurgt, hvad nytte hjemmedatamaterne dog er til. Mit svar er, at de – i skarp kontrast til vores sofakultur – træner intellektet.

For at få maskinen til noget som helst, må man nemlig tænke sig om. Lad os prøve at løse et lille problem. Skriv

```
? 4991 / 9269
```

Vi får 0.538461539, altså med 9 decimalers nøjagtighed. Vi ville imidlertid meget hellere have en forkortet brøk.

Lad os se, går 3 ikke op i begge tal? Næh ... øh ...

Normalt løser man sådan et problem ved at opløse begge tal i primfaktorer. Primtal er tal, kun 1 og tallet selv går op i, så når man har skrevet et tal ud som et gangestykke, hvor alle tal (faktorer) er primtal, kan vi sådan set ikke komme videre. Et simpelt eksempel er 42 / 66, som vi kan få til

$$\frac{2*3*7}{2*3*11} = \frac{\not{2}*\not{3}*7}{\not{2}*\not{3}*11} = \frac{7}{11}$$

Vi har opløst både 42 og 66 i primfaktorer, og det var let nok at se, at både 2 og 3 gik op i både 42 og 66. Men 4991/9269? Her må vi tage maskinen til hjælp.

Vi behøver med andre ord en ordre, der hedder sådan noget som: "Opløs dette tal i primfaktorer!" Sådan en ordre findes naturligvis ikke, vi må lave den som en kompliceret ordre sammensat af andre, et program. Lad os se, hvad vi skal bruge af klodser.

Et nøglebegreb er "gå op i". Hvordan kan maskinen undersøge, om et tal går op i et andet? Efter at have grublet lidt forsøger vi os med

```
10 INPUT a , b : IF a / b = INT(a / b) THEN PRINT
   "ok" : RUN
20 PRINT "nix" : RUN
```

Hvad er det nu, vi har skrevet? Jo, lad os forestille os, at b går op i a. a / b bliver da et helt tal, og INT kan ikke gøre nogen forskel. Hvis b går op i a, er a / b = INT(a / b).

Hvis derimod b *ikke* går op i a, vil a / b efterlade en rest, som INT vil fjerne, hvorfor a / b ikke kan være det samme som INT(a / b). Sætningen IF a / b = INT(a / b) er datamatisk for HVIS b GÅR OP I a. Vores lille program kan altså faktisk undersøge, om b går op i a. Lad os prøve det:

```
? 49 , 7
ok
? 50 , 7
nix
? 51 , 17
ok
?
```

Vi kan nu fremstille følgende program til opløsning i primfaktorer:

```
10 INPUT a : PRINT a ; "=" ; : b = 2
20 IF a / b = INT(a / b) THEN a = a / b : PRINT b ; :
   IF a > 1 THEN 20
30 IF a > 1 THEN b = b + 1 : GOTO 20
40 PRINT : RUN
```

Lad os først prøve vores 51, som programmet gerne skulle få til 3 og 17:

```
? 51
 51 = 3 17
?
```

Nydeligt – men hvordan sytten gjorde den? Lad os prøve at følge med.

Først fik den tallet 51 og skrev 51 =. Så bliver b sat lig 2.

Hvorfor? Jo, b er jo det tal, vi dividerer med, og vi starter med 2 (vi ved godt, at 1 går op).

Nu undersøger maskinen, om 51 / 2 = INT(51 / 2). Det er det ikke, så den går videre i næste linje. Hvis a er 1, er vi selvfølgelig færdige med opgaven, men det er vi ikke endnu, så vi lader 2 blive en større og går så tilbage til 20.

Nu er det 51 / 3 = INT(51 / 3), det gælder, og denne gang har vi bid. Vi kan altså roligt skrive 3 ud som primfaktor og dividere tallet op i de 51, hvilket giver os 17, som stadig ikke er 1, så vi fortsætter i samme linje – det kunne jo være, 3 gik op en gang til. Så heldige er vi ikke, 17 / 3 er absolut ikke det samme som INT(17 / 3), og vi må ned i 30 og gøre b en større, så vi kan forsøge os med 17 / 4, etc. etc.

Til sidst når vi på denne vis frem til 17, og 17 / 17 er vitterlig lig med INT(17 / 17), nemlig *1*, vi skriver de 17 ud og konstaterer samtidig fornøjet, at opgaven er løst. Og lad os så give den noget at arbejde med! Først vores lille problem fra før:

```
? 4991
 4991 = 7 23 31
? 9269
 9269 = 13 23 31
?
```

og 4991 / 9269 er altså 7 / 13! Det var jo ingen sag. Prøv

```
? 123456789
 123456789 = 3 3 3607 3803
```

Se, det havde jo lige godt taget nogen tid med papir og blyant! Primtal er nogle sjove størrelser.

Kunne vi ikke lave et program, der skrev en stribe af dem op, f.eks. alle primtal fra 1 til 2000? Vi kan jo starte med at lave et program, der undersøger, om et tal er et primtal eller ej:

```
10 INPUT "Tal"; a : b = 2
20 IF a / b = INT(a / b) THEN 50
30 b = b + 1 : IF b < a THEN 20
40 PRINT a ; "er et primtal" : RUN
50 PRINT a ; "er ikke et primtal" : RUN
```

Lige så snart et tal går op, er a diskvalificeret som primtal, og det får vi at vide i 50. Så længe vi ikke har fundet et tal, der går op, må vi cykle rundt i 20-30, indtil a = b, og vi endegyldigt kan konstatere, at a er et primtal. Vi prøver:

```
Tal ? 567
 567 er ikke et primtal
Tal ? 568
 568 er ikke et primtal
Tal ? 569
 569 er et primtal
```

```
Tal ? 570
 570 er ikke et primtal
Tal ?
```

Vi skal med andre ord bruge en løkke uden om løkken, hvor a gennemløber tallene 1–2000. Det kan jo blive ret så kompliceret, eller rettere: det var blevet det, hvis ikke FOR-NEXT-løkken var opfundet. Kan du huske

```
10 PRINT "AMSTRAD" : a = a + 1 : IF a < 10 THEN
   10
```

som vi var så stolte af. Den kan vi nu klare endnu enklere:

```
10 FOR a = 1 TO 10 : PRINT "AMSTRAD" :
   NEXT a
```

I virkeligheden er det præcis det samme, der sker. Maskinen checker, hver gang den har skrevet "AMSTRAD", i NEXT a, hvad a er, og hvis det er mindre end de 10, som er angivet i forbindelse med FOR, skriver den det en gang til. Enklere udtrykt: Maskinen gør hvad som helst, som står mellem FOR og NEXT det antal gange, som er angivet efter FOR. Vi kan også få den til at skrive a:

```
10 FOR a = 1 TO 10 : PRINT a : NEXT
```

Læg mærke til, at vi kan undvære a efter NEXT. Eller prøv

```
10 FOR a = 5 TO 15 : PRINT a : NEXT
```

Vi kan endda få den til at gå mindre end et helt trin ad gangen:

```
10 FOR a = 1 TO 10 STEP 0.5 : PRINT a : NEXT
```

Eller prøv

```
10 FOR a = 10 TO 1 STEP − 1 : PRINT a : NEXT
```

Nu trækker maskinen 1 fra a for hvert gennemløb af løkken i stedet for, og resultatet bliver, at der tælles baglæns. a er ikke nogen helt almindelig variabel. Den kaldes en *kontrolvariabel.* Nu kan vi nemt have to løkker inden i hinanden. Vi kan prøve med den lille tabel:

```
10 FOR a = 1 TO 10 : FOR b = 1 TO 10 : PRINT
   a * b : NEXT b : NEXT a
```

Læg mærke til, at b-løkken skal ligge helt inde i a-løkken. Prøv at bytte om på NEXT b og NEXT a. Maskinen siger

```
Unexpected NEXT in 10
```

det var ikke det NEXT, den havde regnet med. Andre (legale) muligheder er

```
10 FOR a = 1 TO 10 : FOR b = 1 TO 10 : PRINT
   a * b : NEXT : NEXT
```

og

```
10 FOR a = 1 TO 10 : FOR b = 1 TO 10 : PRINT
   a * b : NEXT b , a
```

Kør programmet. Særlig kønt ser det jo ikke ud. Vi kunne også prøve med

```
10 FOR a = 1 TO 10 : FOR b = 1 TO 10 : PRINT
   a * b ; : NEXT b , a
```

Det er stadig ikke nogen rigtig tabel. Vi skulle have mulighed for at bestemme hvert enkelt tals placering, og det får vi med LOCATE. Skriv helt uden for programmet ordren

```
LOCATE 1 , 1 : PRINT "AMSTRAD"
```

Maskinen skriver AMSTRAD øverst til venstre på skærmen. LOCATE 1 , 1 betyder: Skriv næste PRINT på 1. plads på 1. linje – læg mærke til, at pladsen kommer først. Der er altid 25 linjer på skærmen, uanset MODE. MODE 0 giver imidlertid 20 pladser, MODE 1 40, og MODE 2 80. Vi kan nu få

```
10 CLS : FOR a = 1 TO 10 : FOR b = 1 TO
   10 : LOCATE a * 3 , b : PRINT a * b : NEXT b , a
```

Og så er der endda plads til mere:

```
10 MODE 2 : FOR a = 1 TO 15 : FOR b = 1 TO
   20 : LOCATE a * 4 , b : PRINT a * b : NEXT b , a
```

Kunne vi da ikke få hele den store tabel klemt ind? Prøv

```
10 MODE 2 : FOR a = 1 TO 20 : FOR b = 1
   TO 20 : c = a * 4 - 5 : IF a < 6 THEN c = a * 3
20 LOCATE c , b : PRINT a * b : NEXT b , a
```

Hvor der er en vilje, er der en vej! Vores primtalsprogram kunne nu komme til at se cirka sådan ud:

```
10 FOR a = 2 TO 2000 : FOR b = 2 TO SQR(a) : IF
   a / b = INT(a / b) THEN 30
20 NEXT : PRINT a ;
30 NEXT
```

Som du ser, har FOR-NEXT-løkken gjort det hele forholdsvis simpelt. Vi prøver tallene 2–2000 (1 regnes normalt ikke for et primtal) ved at dividere tallene fra 2 til SQR(a) op i dem.

Hvorfor nu SQR? For at spare lidt tid.

Lad os sige, det er 49, vi undersøger. Hvis 7 nu meget mod forventning ikke skulle gå op, gør 8 det i al fald heller ikke, for 7 * 8 = 56.

Hvor længe er maskinen egentlig om opgaven? Jeg får det til knap 3 minutter. En variant er

```
10 INPUT "Primtal mellem"; a : INPUT "og"; b :
   IF a<=2 THEN a=2
20 FOR d=a TO b : FOR e=2 TO SQR(d) : IF
   d / e=INT(d / e) THEN 40
30 NEXT : PRINT d ; : c=c+1
40 NEXT : PRINT : PRINT "Mellem"; a ; "og"; b ;
   "findes" : PRINT c ; "primtal" : RUN
```

Her kan vi få opgivet (og talt) alle primtal i et interval, vi selv vælger, altså f.eks.:

```
Primtal mellem ? 1
og ? 100
2  3  5  7  11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97
Mellem 2 og 100 findes
 25 primtal
Primtal mellem ? 1001
og ? 1100
1009 1013 1019 1021 1031 1033
1039 1049 1051 1061 1063 1069
1087 1091 1093 1097
Mellem 1001 og 1100 findes
```

```
 16 primtal
Primtal mellem ? 1000001
og ? 1000100
1000003 1000033 1000037 1000039
1000081 1000099
Mellem 1000001 og 1000100 findes
 6 primtal
```

Som du vil forstå, egner FOR-NEXT sig storartet til "rutine-opgaver", hvor det samme skal gøres igen og igen med mindre afvigelser. Tag f.eks. det matematiske begreb fakultet.

3 fakultet (skrives gerne 3!) er 6, fordi 1 * 2 * 3 * = 6. 5 ! er af forståelige årsager 120, og så videre. Bortset fra, at vi nu igen må tage maskinen til hjælp:

```
10 a=1 : INPUT b : FOR c=1 TO b : a=a*c :
   NEXT : PRINT b ; "! (fakultet)=" ; a : RUN

? 6
66 ! (fakultet) = 720
? 7
 7 ! (fakultet) = 5040
? 10
 10 ! (fakultet) = 3628800
? 12
 12 ! (fakultet) = 479001600
?
```

Ja, det går hurtigt deropad! 15! er 1.307.670.000.000, og 20! 2.432.900.000.000.000.000. Hvor højt kan vi komme op, før maskinen opgiver?

Ikke mindst er FOR-NEXT-løkken fænomenal til strengdeling, som er grundlaget for tekstbehandling, hvor det er strengene, der kommer i ilden. Det handler kapitel 4 om.

4. STRENGDELING OG DIMENSIONERING

LEN – LEFT$ – RIGHT$ – MID$ – STR$ – VAL – TIME – DIM – ERASE – CHR$ – ASC

Kan du huske

```
10 INPUT "MARS' DRABANTER"; a$ , b$: IF a$=
   "phobos" AND b$="deimos" OR a$="deimos"
   AND b$="phobos" THEN END
20 RUN
```

Den lange linje skyldtes jo, at vi ikke kunne vide, om eleven ville svare "phobos" eller "deimos" *først*. Men tag så f.eks.

```
10 INPUT "STØRSTE FUGL"; a$: IF a$<>"struds"
   THEN RUN
```

Sæt nu, der svares "en struds" eller "strudsen" eller "strudse"? Ja, så er svaret "forkert".

Det ville kræve alt for mange ANDer at dække alle mulige korrekte besvarelser. Problemet er jo, at maskinen ikke virkelig forstår, hvad vi siger.

Hvordan forstår vi da selv svar som ovenstående som rigtige? Jo, vi ser efter, om de indeholder "struds". Det kan maskinen også gøre. Til det brug har den nemlig en række funktioner:

LEN(a$) betyder antallet af karakterer i strengen a$
LEFT$(a$, a) betyder de a første karakterer i strengen a$
RIGHT$(a$, a) betyder de a sidste karakterer i strengen a$

MID$(a$, a , b) betyder b karakterer fra strengen a$ begyndende med karakter nummer a

I det sidste tilfælde kan vi underforstå b. Så betyder

MID$(a$, a) streng a$ fra og med karakter nummer a, og strengen ud.

Læg mærke til, at

MID$(a$, a , 1)

åbenbart betyder: Karakter nummer a i strengen a$! Lad os prøve funktionerne på strengen

a$="KØBENHAVN"

Vi får nu f.eks.

LEN(a$)=9

LEFT$(a$, 3)="KØB"

RIGHT$(a$, 4)="HAVN"

MID$(a$, 4 , 2)="EN"

MID$(a$, 4)="ENHAVN"

MID$(a$, 4 , 1)="E"

Eller prøv

? LEFT$(a$, 5)+RIGHT$(a$, 3)

```
10 a$="KØBENHAVN": FOR a=1 TO 8: PRINT
   MID$(a$, a ,2): NEXT
```

skriver alle mulige *delstrenge* af a$ på 2 karakterer ud. Laver vi dette program om til

```
10 a$="KØBENHAVN": FOR a=1 TO 8: IF MID$
   (a$, a , 2)="EN" THEN PRINT "ok"
20 NEXT
```

får vi et program, der checker, om en af disse delstrenge er "en". Vi kan nu lave vores strudseprogram om til

```
10 INPUT "STØRSTE FUGL"; a$: FOR a=1 TO
   LEN(a$): IF MID$(a$, a , 6)="struds" THEN
   END
20 NEXT: RUN
```

Vi kan også gå den anden vej og lave et program, der kan konstatere forekomsten af forskellige delstrenge i strengen "KØBENHAVN". Prøv

```
10 a$="KØBENHAVN": INPUT "Delstreng"; b$:
   FOR a=1 TO 9: IF MID$(a$, a , LEN(b$))=b$
   THEN PRINT "ok"
20 NEXT
```

eller helt generelt

```
10 INPUT "Streng"; a$: INPUT "Delstreng"; b$:
   FOR a=1 TO LEN(a$): IF MID$(a$, a , LEN(b$))
   =b$ THEN PRINT "ok"
20 NEXT
```

Hvad gør

```
10 INPUT a$: FOR a = 1 TO LEN(a$): IF MID$
   (a$, a, 1) = "a" THEN b = b + 1
20 NEXT: PRINT b
```

Eller prøv

```
10 INPUT "Tekst"; a$: FOR a = 1 TO LEN(a$): IF
   MID$(a$, a, 8) = "computer" THEN
   a$ = LEFT$(a$, a - 1) + "datamat" + RIGHT$
   (a$, LEN (a$) - a - 7)
20 NEXT: PRINT a$
```

Programmet undersøger a$ for forekomsten af delstrengen "computer" og skifter den ud, hvor den forekommer, med det mere danske "datamat". Generelt kan vi have

```
10 INPUT "Tekst"; a$: INPUT "Ret"; b$: INPUT
   "til"; c$: FOR a = 1 TO LEN (a$): IF MID$
   (a$, a, LEN (b$)) = b$ THEN a$ = LEFT$ (a$, a - 1)
   + c$ + RIGHT$(a$, LEN (a$) - a - LEN(b$)+1)
```

Vi kan også bruge funktionerne til lidt mere utraditionelle ting. Prøv

```
10 INPUT "Ord"; a$: FOR a = LEN (a$) TO 1 STEP
   - 1 : b$ = b$ + MID$(a$, a, 1): NEXT : a$ = b$:
   PRINT a$: RUN
```

Nemlig, programmet bakker snagvendt:

```
Ord ? amstrad
dartsma
```

```
Ord ? dette er en datamat
tamatad ne re etted
Ord ? en af dem der red med fane
enaf dem der red med fa ne
Ord ?
```

Eller prøv

```
10 MODE 0 : INPUT "Tekst" ; a$: FOR a = 1 TO
   20 – LEN (a$) : a$= a$+ "m" : NEXT : CLS
20 LOCATE 1 , 13 : PRINT LEFT$(a$, 20) : a$=
   RIGHT$(a$, LEN(a$) – 1) + LEFT$(a$, 1) : FOR
   a = 1 TO 50 : NEXT : GOTO 20
```

Kør programmet og indtast f.eks. sætningen

* Her er AMSTRAD lysavis med de seneste nyheder *

En lysavis er nemlig lige præcis, hvad det er, hvilket du vil opdage, når du sætter den i gang med ENTER. Men hvordan virker den? For det første giver MODE 0 os gode store bogstaver. Så får vi teksten ind, f.eks.

KONSTANTINOPOLITANERINDE

Sæt, det nu bare havde været KONSTANTIN? Så var den ikke gået, for vi skal bruge en streng på mindst 20 karakterer (den skal jo fylde skærmen ud).

Kan du se, hvad FOR-NEXT-løkken i 10 gør? Nemlig, den fylder ud med så mange mellemrum som nødvendigt.

Så sletter vi skærmen en gang for alle og går til biddet. De første 20 karakterer af vores streng skrives midt på skærmen)

KONSTANTINOPOLITANER

Men nu bliver a$ noget andet, nemlig ved, at strengens første karakter flyttes om bagi, altså

ONSTANTINOPOLITANERINDEK

Kan du se, hvad FOR-NEXT i 20 gør? Nemlig, det er kun en pause, som vi kan forkorte eller forlænge, som vi vil. Så bliver vi sendt tilbage og tager 20 igen, maskinen skriver

ONSTANTINOPOLITANERI

og i næste omgang

NSTANTINOPOLITANERIN

og i næste igen

STANTINOPOLITANERIND

Virkningen er som af en tekst, der ruller over skærmen. Alt dette bygger naturligvis på maskinens behagelige evne til at "addere" bogstaver. Hvad med cifrene i et tal, så vi f.eks. kunne få et program, der udregnede tværsum? Prøv

```
10 INPUT "TAL"; a$: FOR a=1 TO LEN(a$):
   b=b+MID$(a$, a, 1): NEXT: PRINT b
```

Maskinen skriver

Type mismatch in 10

Vi har jo forsøgt at lægge en streng sammen med et tal. Det, vi skal bruge, er en funktion, der laver et tal om til en streng, og tilbage igen. En sådan funktion har vi i

STR $

og

VAL

Vi kan nu få

```
10 INPUT "TAL"; a : a$= STR$(a) : FOR a = 2 TO
   LEN (a$) : b = b + VAL (MID$(a$, a , 1)) : NEXT :
   PRINT b
```

Hvorfor *2* TO LEN (a$)? Fordi den første karakter i et tal altid er et "fortegn".

Når tallet er positivt, er det underforstået (+), men er det negativt (under 0; når vi skylder penge væk), kommer det frem. Der er i alle tilfælde ingen grund til, at maskinen "vurderer" et mellemrum. Vi kan nu lave endnu flere spil:

```
10 MODE 0
20 a$= a$+ MID$(STR$(INT (RND * 10)) , 2) : a = a + 1 :
   PRINT : PRINT a$: FOR b = 1 TO 500 + a * 100 :
   NEXT : CLS : INPUT "Tal" ; b$: IF b$= a$ THEN
   20
30 PRINT : PRINT "Nej, "; a$ : PRINT : PRINT
   a * 100 ; "points" : PRINT
```

Maskinen opgiver et tal på et enkelt ciffer, hvorefter det forsvinder fra skærmen. Den beder så om at få det gentaget.

Kan vi det, føjer den et ciffer til, og så fremdeles. Læg mærke til, at vi får lidt mere tid pr. gæt. Hvor meget, præcis? Prøv

```
10 FOR a = 1 TO 100000 : NEXT
```

og se, hvor længe den varer. Ja, nu burde maskinen have et ur. Når du er færdig med spillet, kan du slukke maskinen og tænde igen og derefter taste

```
? TIME
```

Du får måske 3000. Dividerer du nu dette tal med 300, får du det antal sekunder, maskinen har været tændt.

Hvordan starter jeg nu uret forfra igen? Det kan du kun ved at slukke for maskinen. Men skidt med det. Prøv

```
10 a=TIME : FOR b=1 TO 100000 : NEXT : PRINT
   (TIME-a) / 300 ; "sekunder"
```

Enkelt, ikke sandt? Knap 109 sekunder. Som på de fleste maskiner svarer en FOR-NEXT-løkke på 1000 "omdrejninger" cirka til 1 sekund. Lidt mere avanceret er

```
10 FOR a=1 TO 4 : a$=a$ +MID$(STR$(INT
   (RND*10)), 2) : NEXT
20 c=c+1 : PRINT : PRINT c ; ". FORSØG" ; : INPUT
   b$: d=0 : e=0 : FOR a=1 TO 4 : IF MID$(b$, a ,
   1)=MID$(a$, a , 1) THEN d=d+1 : GOTO 50
30 FOR b=1 TO 4 : IF MID$(b$, a , 1)=MID$(a$,
   b , 1) THEN e=e+1 : GOTO 50
40 NEXT
50 NEXT : PRINT : PRINT "RIGTIGE CIFRE
   RIGTIGT PLACEREDE :" ; d : PRINT "RIGTIGE
   CIFRE FORKERT PLACEREDE :" ; e : IF b$=a$
   THEN PRINT : END
60 GOTO 20
```

Vi prøver: Maskinen tænker på et tal på fire cifre, som vi skal gætte:

```
  1 . FORSØG ? 1234

RIGTIGE CIFRE RIGTIGT PLACEREDE : 0
RIGTIGE CIFRE FORKERT PLACEREDE : 2

  2 . FORSØG ? 5678

RIGTIGE CIFRE RIGTIGT PLACEREDE : 0
RIGTIGE CIFRE FORKERT PLACEREDE : 1

  3 . FORSØG ? 9012

RIGTIGE CIFRE RIGTIGT PLACEREDE : 0
RIGTIGE CIFRE FORKERT PLACEREDE : 3

  4 . FORSØG ? 7890

RIGTIGE CIFRE RIGTIGT PLACEREDE : 0
RIGTIGE CIFRE FORKERT PLACEREDE : 1
```

7890 og 1234 giver til sammen det samme antal rigtige som 9012. Hverken 3, 4, 7 eller 8 er altså med i tallet. Tilbage står 0, 1, 2, 5, 6 og 9.

Af 1. forsøg følger, at 1 og 2 er rigtige. Af 9 og 0 er en rigtig, men ikke begge, og af 5 og 6 er en rigtig, men ikke begge.

```
  5 . FORSØG ? 6789

RIGTIGE CIFRE RIGTIGT PLACEREDE : 0
RIGTIGE CIFRE FORKERT PLACEREDE : 0
```

Tallet består altså af cifrene 0-1-2-5. Mulige placeringer er

```
0 – 0 –
– 1 – 1
2 – 2 –
– 5 5 5
```

eller rettere

```
– 5 – 5
```

idet, hvis 5 stod på 3. plads, 0 og 2 begge måtte stå på 1. plads. Vi gætter på

```
 6 . FORSØG ? 0125

RIGTIGE CIFRE RIGTIGT PLACEREDE : 0
RIGTIGE CIFRE FORKERT PLACEREDE : 4
```

Løsningen må nu nødvendigvis være

```
 7 . FORSØG ? 2501

RIGTIGE CIFRE RIGTIGT PLACEREDE : 4
RIGTIGE CIFRE FORKERT PLACEREDE : 0
```

Prøv at se, hvor få forsøg du kan komme ned på. Som du ser, kan selv et uhyre enkelt spil (6 linjer!) være både en udfordring og en fornøjelse.

Mest får vi dog brug for strengdelingen i databaser. For at kunne lave sådan nogle, må vi imidlertid vide lidt om *sæt.*

Hvad er så et sæt? Et sæt er et system af variabler, hvis navne kun adskiller sig numerisk fra hinanden, hvilket giver os mulighed for at behandle dem helt anderledes effektivt.

```
10 FOR a = 1 TO 10 : a (a) = 37 : NEXT
```

tilskriver på en og samme gang værdien 37 til ti forskellige variabler, som vi senere kan bruge som a(1),a(2),...a(10). Men prøv så at køre

```
10 FOR a=1 TO 100 : a(a)=37 : NEXT
```

Vi får

```
Subscript out of range in 10
```

Fejlmeddelelsen betyder "for stort indeks". Indeks er tallet i parentesen.

Det er klart, at vi på en gang forlanger en forfærdelig masse plads af maskinen, når vi i en enkelt linje vil tilskrive 100 variabler. Skal vi derfor danne sæt på over 11 *indekserede variabler*, må vi først gøre maskinen opmærksom på, at vi kommer til at behøve denne plads. Det gør vi med ordren

```
DIM
```

Vi kan så have

```
10 DIM a( 100 ):FOR a=1 TO 100 : a( a )=37 :
   NEXT
```

og checke med

```
? a( 77 )
```

Det fungerer. Faktisk har vi med *dimensioneringsordren* skaffet plads til *101* variabler, idet den første variabel i et sæt altid har indeks 0! Vi kan også dimensionere *strengsæt*. Vi kunne altså sagtens have haft

```
10 DIM a$( 100 ): FOR a = 1 TO 100 : a$( a )=
    "KØBENHAVN": NEXT
```

Endelig kan vi danne sæt i flere dimensioner. Prøv

```
DIM b( 15 , 15 )
```

Vi har nu 16 * 16 = 256 variabler til rådighed med navnene

```
b( 0 , 0 ), b( 0 , 1 ),b( 0 , 2 ), . . . b( 0 , 15 ), b( 1 , 0 ), . . .
b( 15 , 15 )
```

Prøv nu

```
DIM b( 15 )
```

Vi får

```
Array already dimensioned
```

På denne måde forhindrer maskinen os i at slette et gammelt sæt med et nyt ved en fejltagelse. Prøv nu

```
ERASE b
```

Nu er det gamle sæt slettet, og vi kan dimensionere det ny:

```
DIM b( 15 )
```

Læg mærke til, at a (77) stadig er 37. Skulle vi have slettet begge, havde det heddet

```
ERASE a , b
```

En mand ved navn CONWAY opstillede engang et sæt (temmelig forenklede) leveregler for levende celler. Han gik ud fra en "flad" koloni, som den kunne brede sig over en glasplade, altså

```
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
```

Som det vil fremgå, har hver celle, bortset fra de yderste, 8 naboer – dette er selvfølgelig efter dette system det største antal naboer, en celle kan have. I kolonien

```
 0
00
```

har hver celle 2 naboer. *Conways regler* siger nu:

1. En celle med 2 eller 3 naboer overlever til næste generation.
2. En celle "fødes" i *næste generation* i et felt, der er tomt i *denne generation*, hvis dette tomme felt har nøjagtig 3 naboer.

Underforstået i 1. er, at en celle med *under* 2 eller *over* 3 naboer dør. Dette skal vi nu forsøge at lave om til et program: *Life*. Vi prøver

```
10 MODE 2:DIM a(25,82),b(25,82):a=-1:
   FOR b=2 TO 24:FOR c=2 TO 81:IF
   RND<0.5 THEN b(b,c)=1
20 NEXT c,b
30 a=a+1:LOCATE 1,1:PRINT "Generation";a:
   d=0:FOR b=2 TO 24:FOR c=2 TO
   81:a(b,c)=b(b,c):LOCATE c-1,b:IF
   a(b,c)=1 THEN PRINT "O";:d=d+1
40 IF a(b,c)=0 THEN PRINT "m";
50 NEXT c,b:LOCATE 1,25:PRINT "Population"
   ;d:FOR b=2 TO 24:FOR c=2 TO 81:d=0:
   d=d+a(b-1,c-1)+a(b-1,c)+a(b-1,
   c+1)+a(b,c-1)+a(b,c+1)+a(b+1,c-1)
   +a(b+1,c)+a(b+1,c+1):IF a(b,c)=1
   AND d<>2 AND d<>3 THEN b(b,c)=0
60 IF a(b,c)=0 AND d=3 THEN b(b,c)=1
70 NEXT c,b:GOTO 30
```

Lad os nu prøve at køre programmet.

Først ser vi generation 0, som er opstået ved, at vi har sat en celle hver anden gang, og et "mellemrum" hver anden, hvilket er foregået ved, at der for hver plads har været lige store chancer for en celle og et mellemrum (RND<0.5). Nederst har vi populationen, altså det totale antal celler i en generation. Jeg lagde ud med 880.

Først i anden generation begynder Conways regler at manifestere sig, hvorfor vi har kaldt den generation 1. Dette medfører som regel en udtynding, hos mig til 615.

Tilbage står nu blot at vente og kigge, eventuelt forlade maskinen og vende tilbage og se, hvordan det går med kolonien. Det er nærmest som at have en potteplante, og det går trods alt lidt hurtigere, maskinen klarer en generation på knap to og et halvt minut. I 5. generation var min koloni gået ned til 487, i 10. til 414, men stabile former var også begyndt at danne sig. En form som

```
00
00
```

er af forståelige årsager stabil, og det samme er

```
 0
0 0
0 0
 0
```

og

```
 00
0 0
 0
```

ja selv

```
00 00
0   0
 000
```

mens

```
000
```

"blinker" til

```
0
0
0
```

og tilbage igen. Det morsomme er jo, at disse "egnede" livsformer udvikler sig spontant i overensstemmelse med "naturlovene". Du kan så føre zoologibog over dem.

Nogle har måske påtrængende naboer, der "æder" dem. Sådanne "krystaller" kan være voldsomt ekspansive, som du vil se.

Generation 15 gav mig 376 celler, og 20 havde 320. Programmer af denne type kaldes *simulationer*, vi efterligner og studerer et forløb, i dette tilfælde cellevækst.

Vi kan også lave lidt mere avancerede simulationer, men så skal vi først have fat i noget bedre grafik. Prøv

```
10 FOR a=65 TO 90:PRINT CHR$(a);:NEXT
```

Maskinen skriver alle de store bogstaver.

```
10 FOR a=97 TO 122:PRINT CHR$(a);:NEXT
```

giver dig de små. Som du vil forstå, virker a her som en særlig kode, den såkaldte ASCII, hvor hver karakter, maskinen kan skrive, har et særligt tal. Du kan også gå den anden vej og bede om en karakters ASCII, f.eks.

```
? ASC("A")
```

Maskinen svarer 65. Tallene 0-9 har ASCII-koder 48-57, og dermed har vi optaget 26*2+10=62 af de 255 koder – der er med andre ord ikke så lidt at tage af. Meget plads optages naturligvis af diverse tegn. Prøv

```
10 FOR a=32 TO 199:PRINT CHR$(a);:NEXT
```

127-159 er ikke helt almindelige. Men prøv så

```
10 MODE 0:FOR a=1 TO 10:FOR b=1 TO 10:
   LOCATE b*2,a*2:PRINT CHR$(a*10+b+
   145):NEXT b,a
```

eller

```
10 MODE 0 : FOR a = 1 TO 12 : FOR b = 1 TO 3 :
   LOCATE b * 6 - 5 , a * 2 : c = a * 3 + b + 216 : PRINT
   c ; CHR$( c ) : NEXT b , a
```

Vi har her alle de spillekort, noder, eksplosioner, rumskibe, små mænd og bomber, vi skal bruge i et primitivt bang-bang-spil. Hvad med 0-31? De er knap så simple. Prøv

```
PRINT CHR$( 7 )
```

og

```
10 MODE 0 : FOR a = 0 TO 740 STEP 10 : PRINT
   "DYT –" ; CHR$( 7 ) ; : FOR b = 0 TO 740 - a :
   NEXT b , a : RUN
```

Der er givetvis her tale om et af bogens mere intellektuelle programmer. CHR$(7) kan man altså høre, og det er typisk for de lave ASCII-numre, at de ikke har nogen skærmrepræsentation.

Et eksempel er ENTER. Hvordan ser ENTER ud?

ENTER er CHR$(13). Det skal vi senere få brug for. Vi har nu de fornødne særlige karakterer til vores simulation og kan gå i gang:

```
10 MODE 2 : DIM a ( 25 , 82 ) , b ( 25 , 82 ) : FOR a = 1
   TO 200 : b ( INT ( RND * 23 ) + 2 , INT ( RND * 80 ) +
   2 ) = 1 : NEXT : FOR a = 1 TO 200 : b ( INT ( RND *
   23 ) + 2 , INT ( RND * 80 ) + 2 ) = 9 : NEXT
20 b = b + 1 : LOCATE 1 , 1 : PRINT "Anno" ; b : a = 0
   : c = 0 : FOR d = 2 TO 24 : FOR e = 2 TO 81 :
   a ( d , e ) = b ( d , e ) : LOCATE e - 1 , d : IF a ( d , e )
   = 0 THEN PRINT "m"
```

```
30 IF a(d,e)=1 THEN PRINT CHR$(248);CHR$
   (7):a=a+1
40 IF a(d,e)=2 THEN PRINT CHR$(244);CHR$
   (7):a=a+1
50 IF a(d,e)>2 AND a(d,e)<9 THEN PRINT
   CHR$(143);CHR$(7):a=a+1
60 IF a(d,e)=9 THEN PRINT CHR$(178):c=
   c+1
70 NEXT e,d:LOCATE 1,25:PRINT "Stammer";
   a;"Hjorteflokke";c:FOR d=2 TO 24:FOR
   e=2 TO 81:IF a(d,e)=0 OR a(d,e)=9
   THEN 140
80 IF a(d-1,e)=9 THEN b(d-1,e)=0:b(d,e)
   =b(d,e)+1
90 IF a(d,e-1)=9 THEN b(d,e-1)=0:b(d,e)
   =b(d,e)+1
100 IF a(d,e+1)=9 THEN b(d,e+1)=0:b(d,
   e)=b(d,e)+1
110 IF a(d+1,e)=9 THEN b(d+1,e)=0:b(d,
   e)=b(d,e)+1
120 IF a(d,e)>3 THEN b(d,e)=a(d,e)-2:
   b(INT(RND*23)+2,INT(RND*80)+2)=2
130 b(d,e)=b(d,e)-1
140 NEXT e,d:FOR d=1 TO c:e=INT(RND*
    23)+2:f=INT(RND*80)+2:IF b(e,f)=0
    THEN b(e,f)=9
150 NEXT:GOTO 20
```

Lad os prøve at køre det. Frem på skærmen toner nærmest et hulebillede med små mænd og tyrehoveder. Det er nu ikke tyre, men hjorteflokke, og de små mænd er (menneske)stammer.

Det hele drejer sig om økologisk balance. Når stammerne er i nærheden af hjorteflokke, æder de dem og formerer sig. Måske holder de blot netop (menneske)bestanden

oppe, eller måske bliver de til små landsbyer (pyramide) eller større bysamfund (massivt kvadrat). I det sidste tilfælde risikerer de overbefolkning, og en ny stamme flytter ud i vildmarken.

Hjortene formerer sig blot ved at fordoble deres antal. Lad os tage en *sample run*, en prøvekørsel.

Første år har vi 168 stammer og 193 hjorteflokke. Det er slet ikke mad nok, så det følgende år dør menneskene som fluer, hvorfor antallet af stammer falder til 64. Det er selvfølgelig godt for hjorteflokkene, der når op på 285.

Det følgende år er der kun 26 stammer tilbage og 488 flokke. Denne udvikling kulminerer det sjette år med kun 8 stammer.

Allerede det næste år har vi 9 på grund af al den gode mad, men der er stadig ikke mennesker nok til at gøre noget ved hjortenes befolkningseksplosion. 8. år har vi 1760 hjorteflokke til 10 stammer, anno 10 er der over 1800.

Så begynder menneskene at brede sig, overalt skyder der samfund frem i madhavet, og der begynder at danne sig små tørre pletter omkring dem. År 15 har vi 66 stammer og 1705 hjorteflokke, år 20 er der 231 stammer.

Endnu er der ikke gjort faretruende indhug på hjortebestanden, men nu begynder det at knibe. År 25 er der kun 839 hjorteflokke til 454 sultne stammer.

År 28 er der 479 stammer, men nu er maden også ved at være sluppet op, og anno 35 er dette tal atter faldet til 348. Hvis du vil have maskinen til at huske tallene for dig, kan du tilføje:

```
5 DIM s(50),h(50)
160 CLS:FOR b=1 TO 50:PRINT b;s(b);h(b),
    :NEXT
```

og ret 150 til

```
150 NEXT : s ( b ) = a : h ( b ) = c : IF  b < 50  THEN  20
```

Derved fik jeg følgende tal, som vi skal få glæde af i et senere kapitel:

Anno	Stammer	Hjorteflokke	
1	168	193	
2	64	285	
3	26	488	
4	14	786	
5	10	1145	
6	8	1476	FÆRREST STAMMER
7	9	1683	
8	10	1760	
9	12	1797	
10	16	1801	FLEST HJORTEFLOKKE
11	21	1799	
12	29	1783	
13	40	1752	
14	53	1739	
15	66	1705	
16	90	1659	
17	112	1605	
18	151	1537	
19	179	1442	
20	231	1364	
21	268	1256	
22	320	1187	
23	360	1062	
24	421	960	
25	454	839	
26	468	772	
27	477	704	
28	479	662	FLEST STAMMER
29	448	666	

30	432	646	FÆRREST HJORTEFLOKKE
31	417	658	
32	397	682	
33	376	727	
34	353	784	
35	348	841	FÆRREST STAMMER
36	354	889	FLEST HJORTEFLOKKE
37	361	878	
38	377	864	LIGEVÆGT OMKRING
39	390	845	400 STAMMER OG 800
40	398	838	HJORTEFLOKKE
41	410	818	
42	412	789	
43	421	794	
44	413	759	
45	414	775	
46	423	779	
47	415	769	
48	412	750	
49	411	716	
50	401	730	

5. DATA

READ – DATA – RESTORE – SAVE – CAT – LOAD – MERGE – RENUM – UPPER$ – LOWER$ – INKEY$ – ABS

Prøv

```
10 MODE 0
20 a=a+1
30 b=INT(RND*9)+1 : IF a(b)=1 THEN 30
40 a$(a)=STR$(b) : a(b)=1 : IF a<9 THEN 20
50 PRINT : PRINT : FOR a=1 TO 9 : PRINT a$(a);
   : NEXT : c=c+1 : PRINT : PRINT : PRINT
   "FORSØG"; c; ": m";
60 INPUT "VEND"; a : IF a<2 OR a>9 THEN
   PRINT : GOTO 60
70 FOR b=1 TO a : b$(b)=a$(a-b+1) : NEXT :
   FOR b=1 TO a : a$(b)=b$(b) : NEXT
80 FOR a=1 TO 9 : IF VAL(a$(a))<>a THEN 50
90 NEXT : PRINT : FOR a=1 TO 9 : PRINT a$(a); :
   NEXT : PRINT : PRINT
```

En sample run kan give

```
 3 5 1 2 6 9 4 7 8
FORSØG 1 : VEND?
```

Det gælder nu for os om at sætte de ni tal i nummerorden i så få forsøg som muligt. I hvert forsøg kan vi vende et valgfrit antal cifre. Skriver vi f.eks. 6, får vi

```
 9 6 2 1 5 3 4 7 8
FORSØG 2 : VEND?
```

Som du ser, er de *første* 6 cifre i 351269478, nemlig 351269, vendt om til 962153, hvilket giver det nye tal 962153478. Vi kan fortsætte

```
 9 6 2 1 5 3 4 7 8
FORSØG 2 : VEND ? 9

 8 7 4 3 5 1 2 6 9
FORSØG 3 : VEND ? 8

 6 2 1 5 3 4 7 8 9
FORSØG 4 : VEND ?
```

Med denne metode kommer tallene på plads et for et, men der er heller ikke meget strategi i det. Men læg mærke til, hvordan vi var så heldige at få to tal på plads i det sidste forsøg. Dette kan inspirere os til en genvej:

```
 6 2 1 5 3 4 7 8 9
FORSØG 4 : VEND ? 6

 4 3 5 1 2 6 7 8 9
FORSØG 5 : VEND ? 2

 3 4 5 1 2 6 7 8 9
FORSØG 6 : VEND ? 3

 5 4 3 1 2 6 7 8 9
FORSØG 7 : VEND ? 5

 2 1 3 4 5 6 7 8 9
FORSØG 8 : VEND ? 2

 1 2 3 4 5 6 7 8 9
```

Med den rigtige taktik vil du se, at du kan klare dig i meget få forsøg.

Læg mærke til, at vi her betjente os af så lille et sæt, at vi ikke behøvede at dimensionere. Det behøver vi heller ikke for at lave en (meget) lille database:

```
10 FOR a=1 TO 10: PRINT a;: INPUT a$(a):
   NEXT
20 PRINT: INPUT "SØGEORD"; a$: PRINT: FOR
   a=1 TO 10: FOR b=1 TO LEN(a$(a)): IF
   MID$(a$(a),b,LEN(a$))=a$ THEN PRINT
   a$(a)
30 NEXT b,a: GOTO 20
```

Indtast nu følgende data:

```
 1 ? alarmering 000
 2 ? erhvervsservice 0048
 3 ? klokken 0055
 4 ? telex 0024
 5 ? sportsavisen 0052
 6 ? telefonavisen 0051
 7 ? telefonhuset 0019
 8 ? vejledning 0030
 9 ? lokal vejrmelding 0053
10 ? landsvejrmelding 0054

SØGEORD? klokken

klokken 0055

SØGEORD? vejrmelding

lokal vejrmelding 0053
landsvejrmelding 0054
```

SØGEORD ?

Læg mærke til det sidste tilfælde. I et kartotek ville vi have slået "vejrmelding" op forgæves. Prøv

SØGEORD ? tele

telex 0024
telefonavisen 0051
telefonhuset 0019

SØGEORD ? avis

sportsavisen 0052
telefonavisen 0051

SØGEORD ?

Men hvad nu, når jeg kører det næste gang? Så slettes jo alle variabler.

Jeg kunne selvfølgelig "snyde" og kun køre det med GOTO 20, men hvad så, når jeg skulle gemme det på bånd? Jeg laver nu en ny version:

```
10 FOR a = 1 TO 10 : READ a$( a ) : NEXT
20 som før
30 som før
40 DATA alarmering 000 , erhvervsservice 0048 ,
   klokken 0055 , telex 0024 , sportsavisen 0052 ,
   telefonavisen 0051 , telefonhuset 0019 , vejledning
   0030 , lokal vejrmelding 0053 , landsvejrmelding
   0054
```

Prøv de samme eksempler på denne udgave. Hvad har vi gjort?

Vi har proppet alle oplysningerne (data) i en DATA-linje med , imellem. Disse data "læser" programmet i 10 ved hjælp af READ. Prøv at tilføje til linje 40:

nummeroplysning 0033

Kør programmet og bed om

SØGEORD ? nummer

Maskinen har selvfølgelig ikke fået det med. Prøv så at fjerne både nummeroplysningen og landsvejrmeldingen. Nu siger maskinen

DATA exhausted in 10

"vi har ikke DATA til sidste skuffe", det vil sige kun, hvis du husker at fjerne *kommaet før* landsvejrmeldingen også, ellers opfatter maskinen det nemlig, som om der *er* et sidste datum, der blot er en tom streng. Så vidt maskinen kan se, er der i linjen

```
100 DATA erwin , elsebeth , , lea , marcus , , , rachel ,
    daniel ,
```

ikke mindre end 10 data, nemlig

erwin
elsebeth
(tom streng)
lea
marcus
(tom streng)
(tom streng)
rachel

daniel
(tom streng)

og i

```
100 DATA 12 , 34 , , , , 567 ,
```

7 data, nemlig

```
12 , 34 , 0 , 0 , 0 , 567 , 0
```

```
RESTORE
```

betyder, at vi skal begynde forfra på DATA mens

```
RESTORE a
```

hvor a er et linjenummer, angiver, at der skal startes forfra i denne datalinje. Prøv

```
10 FOR a = 1 TO 3 : READ a$( a ) : NEXT :
   RESTORE : FOR a = 4 TO 6 : READ a$( a ) : NEXT :
   RESTORE 30 : FOR a = 7 TO 10 : READ a$( a ) :
   NEXT : FOR a = 1 TO 10 : PRINT a$( a ) : NEXT
20 DATA en , to
30 DATA tre , fire
40 DATA fem , seks
```

Som sagt overflødiggør databasen alfabetiserede kartoteker, men vi kan skam også fremstille et program, der alfabetiserer. Udtrykket

```
"A"<"B"
```

giver nemlig udmærket mening for maskinen. Det betyder: As ASCII (65) er mindre end Bs (66), eller simpelt hen: A kommer før i alfabetet. Tilsvarende er

```
"ABC"<"ABE"
```

Vi kan nu have

```
10 DIM a$(100):a$(1)="zz"
20 a=a+1
30 INPUT a$:IF a$="*" THEN 70
40 FOR b=1 TO 100:IF a$<a$(b) THEN 60
50 NEXT
60 FOR c=a TO b STEP -1:a$(c+1)=a$(c):
   NEXT:a$(b)=a$:GOTO 20
70 FOR b=1 TO a-1:PRINT a$(b):NEXT:
   GOTO 30
```

Prøv

```
? hund
? kat
? hest
? ko
? gris
? *
gris
hest
hund
kat
ko
? elefant
? giraf
? zebra
? *
```

elefant
giraf
gris
hest
hund
kat
ko
zebra
?

Et noget anderledes program er

```
10 DIM a$(165),b$(165),c$(25):FOR a=1 TO
   165:READ a$(a),b$(a):NEXT
20 INPUT a$:PRINT:a$=a$+"m":a=2:b=1:
   c=1
30 IF MID$(a$,a,1)="m" THEN c$(b)=MID$(a
   $,c,a-c):b=b+1:c=a+1
40 IF a=LEN(a$) THEN 60
50 a=a+1:GOTO 30
60 FOR a=1 TO b-1:FOR c=1 TO 165:IF c$
   (a)=a$(c) THEN c$(a)=b$(c):GOTO 80
70 NEXT
80 PRINT c$(a);"m";:NEXT:PRINT:PRINT:
   GOTO 20
90 DATA a,en,about,om,after,efter,air,luft,
   am,er,America,Amerika,an,en,and,og,
   are,er,bag,kuffert,ball,bold,banana,banan,
   begin,begynde,best,bedste,big,stor,
   black,sort,book,bog,bookshop,boghandel,
   boy,dreng,bring,bringe,British,britisk,
   brother,broder
100 DATA brown,brun,but,men,by,med,can,
   kan,cannot,kan ikke,car,bil,cat,kat,chair,
   stol,city,by,clock,stueur,coal,kul,coffee,
```

```
    kaffe , colour , farve , come , komme , cup , kop ,
    Danish , dansk , day , dag , Denmark , Danmark ,
    dog , hund , dress , kjole , eight , otte , English ,
    engelsk
110 DATA  evening , aften , family , familie , father ,
    fader , find , finde , fine , fin , firm , firma , fish ,
    fisk , five , fem , four , fire , friend , ven , from , fra,
    garden , have , girl , pige , glass , glas , good , god ,
    good-bye , farvel , half , halv , has , har , have , har,
    he , han , her , hendes , here , her , his , hans
120 DATA  house , hus , how , hvor , hundred ,
    hundrede , I , jeg , if , hvis , in , i , is , er , it , den ,
    jacket , jakke , lady , dame , letter , brev , little , lille ,
    live , bor , long , lang , lunch , frokost , man ,
    mand , many , mange , me , mig , money , penge ,
    morning , morgen , mother , moder , mr , hr ,
    mrs , fru , my , min , name , navn
130 DATA  new , ny , nine , ni , no , nej , not , ikke , of ,
    af , old , gammel , one , en , only , kun , or , eller ,
    our , vor , past , over , pay , betale , play , lege ,
    pound , pund , quarter , kvarter , say , siger ,
    school , skole , schoolbook , skolebog , see , se ,
    seven , syv , she , hun , shop , butik , short , kort ,
    sir , hr , six , seks
140 DATA  soup , suppe , street , gade , Swedish ,
    svensk , table , bord , take , tage , taxi , taxa , tea ,
    te , telephone , telefon , television , fjernsyn , ten ,
    ti , their , deres , them , dem , there , der , they ,
    de , this , denne , three , tre , ticket , billet , time ,
    tid , to , til , today , i  dag , tomorrow , i  morgen
150 DATA  train , tog , trip , tur , trousers , bukser ,
    twenty , tyve , two , to , uncle , onkel , us , os , very ,
    meget , waiter , tjener , walk , spadsere , was , var ,
    water , vand , we , vi , were , var , what , hvad ,
```

```
    where , hvor , white , hvid , who , hvem , wife ,
    kone , will , vil , window , vindue , with , med ,
    work , arbejde
160 DATA yellow , gul , yes , ja , you , du , your , din
```

Prøv at indtaste en dansk sætning inden for ovenstående ordforråd. Som du vil se, leverer maskinen en tilfredsstillende, om ikke altid strengt grammatisk oversættelse.

Med et øget ordforråd kan resultatet blive ret imponerende, og tillige nyttigt, især, hvis det er et sprog, man ikke selv forstår. De første 5000 gloser skulle maskinen snildt kunne indeholde.

Når vi nu en gang for alle har indtastet dem, kan vi sætte maskinen til at oversætte, hvad det skal være. Dette kræver naturligvis, at vi har program komplet med gloseliste på bånd. Lad os prøve at gemme (SAVE) ovenstående.
Sæt bånd i maskinen, idet du sikrer dig, at det er spolet helt tilbage. Du skal nu finde et passende navn til dit program. Lad os tage "ENGELSK". Altså skriver du

```
SAVE "ENGELSK"
```

fulgt af ENTER. Maskinen skriver

```
Press REC and PLAY then any key:
```

og det er præcis, hvad du gør. Tryk altså REC og PLAY ned, så de låser, og tast f.eks. ENTER igen. Maskinen går i gang med at SAVE. Du kan høre ligesom en lav hvinen; det er lydsignalet, som indeholder programmet. Samtidig skriver maskinen

```
Saving ENGELSK block 1
```

og lidt senere

Saving ENGELSK block 2

Maskinen oplagrer altså programmet i "blokke". Til sidst skriver den Ready, og programmet er i kassen.

Tast nu STOP på båndoptageren og REW, så den spoler båndet tilbage til start igen. Hvis du nu trykkede på STOP igen, ville du få båndet ud af maskinen, men måske vil du gerne have en slags bevis på, at alt er gået godt. Tast nu

CAT

Det står for CATalogue eller katalog. Når du nu, som maskinen opfordrer til, trykker på PLAY og en tast på selve datamaten, vil den begynde at remse op, hvad der er på båndet. På denne måde verificerer du optagelsen. Den siger nemlig

ENGELSK block 1 $ Ok

og lidt senere

ENGELSK block 2 $ Ok

Sluk og tænd igen for maskinen og spol igen båndet tilbage til start. Tast nu

LOAD "ENGELSK"

og tryk PLAY ned. Tast f.eks. ENTER, og du får dit program læst ind i maskinen. Sluk og tænd igen, men tast nu

RUN "ENGELSK"

Nu vil programmet gå automatisk i gang i det øjeblik, det er i maskinen. Har du glemt, hvad der er på båndet, og ikke først vil have kataloget, kan du bare skrive

LOAD ”

eller eventuelt

RUN ”

Maskinen vil da blot LOADe/køre det første program på båndet. Det sidste er på den meste software, du køber, ”forkortet” til CTRL-ENTER, hvorfor du blot behøver at taste de to for at få programmet ind.

Disse bånd er som regel ”Protected”, beskyttet, så du ikke kan læse LISTen. Du kan gøre det samme. Sæt et nyt bånd i maskinen. Skriv nu

SAVE ”ENGELSK”, P

Når du har programmet på bånd, kan du prøve at slukke og tænde for maskinen og så få det ind i maskinen igen. Prøv som før

LOAD ”ENGELSK”

Det ser udmærket ud, men prøv at LISTe eller blot køre programmet.

RUN ”ENGELSK”

vil LOADe og køre programmet, men du kan stadig ikke LISTe det. I kataloget (CAT) er et beskyttet program forsynet med % (i stedet for $).

Prøv

```
10 FOR a=1 TO 10:b=INT(RND*165)+1:PRINT:
   PRINT "Hvad betyder ";a$(b);:INPUT a$:IF
   a$=b$(b) THEN c=c+1:GOTO 30
20 PRINT:PRINT "Nej - ";b$(b)
30 NEXT:PRINT:PRINT "Du fik ";c;" ud af 10!":
   PRINT
```

Jamen, er det ikke vores gode gamle undervisningsprogram? Det har bare en lille fejl: Det virker ikke.

Hvis vi knyttede det sammen med en gloseliste, kunne det blive et helt pænt lille engelsk-kursus. Havde vi ikke sådan en et sted?

Men det er jo i et helt andet program. Skidt, vi limer dem sammen. Dertil har vi ordren

```
MERGE
```

Vores oprindelige engelsk-program går imidlertid fra 10 til 160, og vi kan ikke have to linjer 10 eller 20. Hvad gør vi så? Vi renummererer. Prøv at indføje linjen

```
25 PRINT "Skvadderhoved!"
```

Måske ikke netop et bruger-venligt program, men hvad . . . Derimod er vi jo afveget fra vores hidtidige pæne nummerering 10-20-30 . . . Prøv nu

```
RENUM
```

for RENUMber. Tag en ny LIST! Skaf dig af med den uhøflige linje. Der blev et hul – endnu en RENUM stopper det.

Men hvad hjælper det alt sammen? Vi skulle gerne have lavet vores 10-20-30 om til 11-12-13.

Hvorfor lige det? Jo, 10 i det oprindelige program er jo indlæsningen, og den kan vi godt bruge, og helst *før* vores

tre linjer. Vi behøver med andre ord en ordre, der siger sådan noget som

RENUMMERER SÅ FØRSTE LINJENUMMER
BLIVER 11 , OG DE TO FØLGENDE BLIVER 12
OG 13

Den har vi faktisk i

RENUM 11 , 10 , 1

11 er det første nye linjenummer, 10 er det første linjenummer i det gamle program, vi vil have lavet om, og 1 er afstanden mellem de følgende linjenumre. Hvad ville

RENUM 25 , 20 , 15

have givet? Nemlig, linjenumrene 10-25-40. Ikke sandt? Linje 20 bliver til linje 25, og derefter går vi frem med 15, så at 30 bliver til 25 + 15 = 40. Eftersom renummereringen skulle starte med 20, er 10 forblevet uforandret.

Men det var linjenumre 11-12-13, vi skulle have. Vi kan nu MERGE:

MERGE "ENGELSK"

Maskinen LOADer som før. Tag en LIST.

DELETE 20 – 80

RENUM

og du har dit engelsk-program. Eller prøv

```
10 DIM a$(66),b$(66):FOR a=1 TO 66:READ
   a$(a): NEXT:FOR a=1 TO 66:READ b$(a):
   NEXT:PRINT "TAXA":a=INT(RND*66)+1
20 b=INT(RND*66)+1:IF a=b THEN 20
30 PRINT "Du er i omegnen af m";a$(a):PRINT
   "Din kunde skal til m";a$(b):PRINT "Tid:
   ";c;".";d:IF a=b THEN 110
40 INPUT "Retning (N / Ø / S / V)";a$:a$=UPPER
   $(a$):IF a$="N" THEN e=VAL(LEFT$
   (b$(a),2))
50 IF a$="Ø" THEN e=VAL(MID$(b$(a),3,2))
60 IF a$="S" THEN e=VAL(MID$(b$(a),5,2))
70 IF a$="V" THEN e=VAL(RIGHT$(b$(a),2))
80 IF e=0 THEN 30
90 a=e:d=d+10:IF d=60 THEN c=c+1:d=0
100 GOTO 30
110 f=f+1:IF f<3 THEN PRINT "Ny kunde":
    GOTO 20
120 a=500-c*60-d:IF a<0 THEN a=0
130 PRINT a;"points"
140 DATA MØRDRUP,SNEKKERSTEN,
    HUMLEBAEK,ESPERGAERDE,NIVERØD,
    ULLERØD,LILLERØD,SLUTTERUP,
    HØRSHOLM,LYNGE,ALLERØD,BIRKERØD,
    HØSTERKØB,VEDBAEK,BURESØ,LYNGE,
    FARUM,BIRKERØD,SØLLERØD,
    JAEGERSBORG,FARUM,HOLTE,BREDE,
    EREMITAGEN,KNARDRUP,VAERLØSE,
    HARESKOV,SORGENFRI
150 DATA ORDRUP,SMØRUMNEDRE,BALLERUP,
    HJORTESPRING,GLADSAXE,GENTOFTE,
    BALLERUP,EJBY,HUSUM,NØRREBRO,
    VRIDSLØSEMAGLE,HERSTEDVESTER,
    HVISSINGE,VANLØSE,FREDERIKSBERG,
    KLØVERMARKEN,MARBJERG,HEDEHUSENE,
```

```
     TAASTRUP , ALBERTSLUND , BRØNDBY ,
     HVIDOVRE , SYDHAVNEN , SUNDBYØSTER
160  DATA ISHØJ, ISHØJ, BRØNDBY, AVEDØRE,
     TAARNBY , TØMMERUP , KARLSLUNDE ,
     MOSEDE , HUNDIGE , KONGELUNDEN ,
     MAGLEBY , KARLSTRUP , MOSEDE , JERSIE ,
     00020300 , 00000401 , 01040500 , 02000003 ,
     03000600 , 05000900 , 00081100 , 00091207 ,
     06001308 , 00111600 , 07121710 , 08131811 ,
     09141912 , 00002013
170  DATA  00160000 , 10170015 , 11182116 ,
     12192217 , 13202318 , 14002419 , 17222600 ,
     18232721 , 19242822 , 20002923 , 00263000 ,
     21273125 , 22283226 , 23293327 , 24003428 ,
     25310000 , 26323530 , 27333631 , 28343732 ,
     29003833 , 31364000 , 32374135 , 33384236 ,
     34004337 , 00404700 , 35414839 , 36424940
180  DATA  37435041 , 38445142 , 00005243 ,
     00460000 , 00470045 , 39485346 , 40495447 ,
     41505548 , 42515649 , 43525750 , 44005851 ,
     47546000 , 48556153 , 49560054 , 50570055 ,
     51586256 , 52006357 , 00606400 , 53616559 ,
     54000060 , 57630000 , 58000062 , 59656600 ,
     60000064 , 64000000
```

Lad os prøve at spille det:

```
TAXA
Du er i omegnen af BIRKERØD
Din kunde skal til BALLERUP
Tid 0 . 0
Retning  (N / Ø / S / V) ?
```

Ja, det må da i hvert fald være sydpå. Vi skriver S og er i HOLTE. Spillet fortsætter

```
Du er i omegnen af HOLTE
Din kunde skal til BALLERUP
Tid 0 . 10
Retning (N / Ø / S / V)? S
Du er i omegnen af HARESKOV
Din kunde skal til BALLERUP
Tid 0 . 20
Retning (N / Ø / S / V)? S
Du er i omegnen af HJORTESPRING
Din kunde skal til BALLERUP
Tid 0 . 30
Retning (N / Ø / S / V)? Ø
Du er i omegnen af GLADSAXE
Din kunde skal til BALLERUP
Tid 0 . 40
Retning (N / Ø / S / V)?
```

Hov, det var vist forkert – der tabte vi tid og points! Men vi bliver bedre til det efterhånden, det vil sige bedre til Københavns eller en hvilken som helst anden geografi, vi måtte ønske at lære.

Læg mærke til UPPER$(a$). Takket være den er det ligegyldigt, om du taster n eller N, eftersom både

```
UPPER$("n")
```

og

```
UPPER$("N")
```

er N. Prøv tilsvarende

```
? LOWER$("n"); LOWER$("N"); LOWER$
("AMSTRAD")
```

En anden lettelse af indtastning er

INKEY$

Prøv

```
10 a$=INKEY$:IF a$="" THEN 10
20 PRINT a$
```

Læg mærke til, at maskinen venter på din indtastning, men at du sparer ENTER. Det kan også give dig en helt anderledes kontrol med inputtet. Tag f.eks.

```
10 INPUT a : IF a<1 OR a>9 THEN 10
```

Men sæt nu, brugeren taster A, f.eks. ved en fejltagelse. Så får han skiltet

```
? Redo from start
```

som han måske ikke forstår et ord af. Men prøv så

```
10 a$= INKEY$: IF a$="" THEN 10
20 a=ASC(a$): IF a<49 OR a>57 THEN 10
30 PRINT "OK -"; a-48
```

INKEY$ behøver imidlertid ikke at vente, og det er blandt andet fidusen i skærmprogrammer med flyvende tallerkener, etc. Prøv

```
10 IF INKEY$="m"THEN PRINT "AMSTRAD";
20 GOTO 10
```

Bemærk forskellen på "" og "m"! Vi kan bruge funktionen i en lille reaktionstest:

```
10 MODE 0: PRINT "Tast ENTER, hvis du ser en
   stjerne HER:";: FOR a=1 TO 1000+RND*1000:
   IF INKEY$ <> ""THEN PRINT"DU SNØD!"
20 NEXT: PRINT "*": a=TIME
30 IF INKEY$="" THEN 30
40 b=TIME: PRINT "Din reaktionstid er";(b-a)
   / 300; "sek": PRINT
```

Du kan også bruge programmet som spiritusprøve for dine gæster:

```
40 b=TIME: c=(b-a) / 300: PRINT "Din
   reaktionstid er; c; "sek": PRINT: IF c>0.2 THEN
   PRINT "TAG EN TAXA!!!": PRINT
```

Prøv også

```
10 CLS: PRINT "DU FØRER RUMSKIBET
   AMSTRAD": PRINT: a=1000
20 b=1000+RND*1000: c=0: d=0: PRINT
   "FJENDTLIGT RUMSKIB I SIGTE"
30 LOCATE 1,4: PRINT "POINTS"; e: PRINT:
   PRINT "AFSTAND"; b: PRINT "RELATIV
   HASTIGHED"; f; "m": PRINT "ENERGI"; a:
   PRINT "KRAFTSKJOLD"; g; "m": PRINT
   "SKADE I SKIB"; INT(h): PRINT: PRINT
   "SKADE I ANDET SKIB"; INT(c)
40 IF RND<0.9 THEN LOCATE 1,14: PRINT
   "FJENDEN SKYDER!!!!!": h=h+RND*
   (1000-b-g) / 10
50 LOCATE 1,16: PRINT "1. Hastighed": PRINT
   "2. Kraftskjold": PRINT "3. Laser": PRINT
```

```
     "4 . Faser" : PRINT : PRINT "Ordre (0 = ingen
     ordre) ?"
 60  a$= INKEY$ : IF a$ = "" THEN 60
 70  i = ASC(a$) − 48 : IF i<0 OR i>4 THEN 60
 80  LOCATE 1 , 23 : IF i = 1 THEN INPUT
     "Hastighed (−100 − 100)" ; f
 90  IF i = 2 THEN INPUT "Kraftskjold (0 − 100)" ; g
100  IF i = 3 THEN a = a − 10 : c = c + RND*(1000 − b)
     / 10
110  IF i = 4 THEN a = a − 20 : c = c + RND*(1000 − b)
     / 5
120  LOCATE 1 , 23 : PRINT "mmmmmmmmmmmmm
     mmmmmmmmmmmmmmmmmm" : a = a − (ABS(f)
     + g)/ 10 : b = b − f − RND * 100 : IF c<0 THEN c = 0
130  IF h<j THEN h=j
140  IF c>= 100 THEN LOCATE 1 , 1 : PRINT
     "FJENDEN ØDELAGT ! ! ! ! ! mmmmmmm" : e = e +
     (25 − d)* 100 : j = h : PRINT : PRINT "NYTm" ; :
     GOTO 20
150  LOCATE 1 , 25 : IF a<= 0 THEN PRINT
     "ENERGI 0" : GOTO 190
160  IF b< =0 THEN PRINT "SAMMENSTØD" :
     GOTO 190
170  IF h> = 100 THEN PRINT "AMSTRAD
     ØDELAGT" : GOTO 190
180  d = d + 1 : GOTO 30
190  GOTO 190
```

Læg mærke til, at vi kan flyve med "negativ relativ hastighed", altså flyve væk fra fjenden. Det giver os imidlertid problemer i 120, hvor vi skal beregne, hvor meget energi vi bruger. En omgang med hastighed 50 koster 5 energienheder, men det betyder jo, at hvis vi bevæger os bort fra det andet rumskib med denne hastighed, får vi 5 enheder foræret! Løsningen er

ABS

Prøv

```
? ABS(37); ABS(-37)
```

ABS (ABSolut værdi) ser bort fra fortegnet. Prøv at køre det:

```
DU FØRER RUMSKIBET AMSTRAD

FJENDTLIGT RUMSKIB I SIGTE
POINTS 0

AFSTAND 1271.94066
RELATIV HASTIGHED 0
ENERGI 1000
KRAFTSKJOLD 0
SKADE I SKIB 0

SKADE I ANDET SKIB 0

FJENDEN SKYDER!!!!!

1. Hastighed
2. Kraftskjold
3. Laser
4. Faser

Ordre (0=ingen ordre)?
```

Vi taster 1 og får

```
Hastighed (-100 - 100)?
```

og kan f.eks. indtaste 50. 2 giver os mulighed for at ændre kraftskjoldets intensitet (og energiforbrug), og 3 og 4 affyrer vores våben, når det andet skib er på skudhold. God jagt!

6. GRAFISK AFBILDNING

GOSUB – RETURN – TRON – TROFF – TAB – ON – PLOT – DRAW – SIN – PI – COS – MOVE – PLOTR – DRAWR – MOVER – INKEY

```
 10 FOR a=1 TO 6:READ a$(a):a(a)=100:
    NEXT:a=1000:MODE 0
 20 CLS:PRINT "ANTAL AKTIE";TAB (17);
    "KURS":FOR b=1 TO 6:PRINT b(b);TAB
    (6);b;a$(b);TAB (16);a(b);:NEXT:PRINT:
    PRINT:PRINT "KAPITAL:";a:PRINT:INPUT
    "KØB / SALG (K / S)";a$:a$=UPPER$(a$):IF
    a$="K" THEN 50
 30 IF a$="S" THEN 70
 40 GOTO 90
 50 GOSUB 130:IF a(b)*c>a THEN 50
 60 b(b)=b(b)+c:a=a-a(b)*c:GOTO 90
 70 GOSUB 130:IF c>b(b) THEN 70
 80 b(b)=b(b)-c:a=a+a(b)*c
 90 FOR b=1 TO 6:a(b)=a(b)+INT(RND*21)-10:
    IF a(b)<10 THEN a(b)=10
100 NEXT:a=a-10:IF a<0 THEN PRINT:
    PRINT "DU ER FALLIT":PRINT:END
110 IF a>10000 THEN PRINT:PRINT "DU ER
    BØRSMATADOR":PRINT:END
120 GOTO 20
130 PRINT:INPUT "AKTIENUMMER";a$:IF
    ASC(a$)<49 OR ASC(a$)>54 THEN 130
140 b=VAL(a$):PRINT:INPUT "HVOR MANGE";
    c:RETURN
150 DATA Jensen, Olsen, Klausen, Fedesen,
    Tomsen, Borgen
```

Læg mærke til linjerne 50 og 70. Hvad er GOSUB ? Prøv

```
 10 PRINT "SPEGE";
 20 GOSUB 100
 30 PRINT "RULLE"
 40 GOSUB 100
 50 PRINT "KØD";
 60 GOSUB 100
 70 PRINT "LEVER";
 80 GOSUB 100
 90 END
100 PRINT "PØLSE"
110 RETURN
```

En kørsel giver

```
SPEGEPØLSE
RULLEPØLSE
KØDPØLSE
LEVERPØLSE
```

men hvordan går det til? 100-110 er en subrutine. Det er noget, maskinen skal gøre mange gange, men forskellige steder i programmet.

GOSUB "spiller pladen". Kunne GOTO ikke have gjort det?

Nej, for hvor skulle et GOTO i 110 have sendt maskinen tilbage til, 30, 50, eller måske 70! RETURN betyder: Fortsæt, hvor du slap (i hovedprogrammet). Prøv

```
TRON
```

og kør programmet igen. Maskinen siger

(10) SPEGE (20)(100) PØLSE
(110)(30) RULLE (40)(100) PØLSE
(110)(50) KØD (60)(100) PØLSE
(110)(90)

Det betyder:

Først tager jeg linje 10 og skriver SPEGE.
Så tager jeg linje 20, der sender mig til linje 100, hvor jeg skriver PØLSE.

Så tager jeg linje 110, der sender mig tilbage til linje 30, hvor jeg skriver RULLE.
Etc., Etc.

Maskinen fortæller hele tiden, hvilken linje den er i gang med. Det kaldes TRACKING, og TRON står for TRacking ON. Prøv

TROFF

der ganske rigtigt står for TRacking OFF. Hvad laver END i linje 90? Prøv at droppe det. Maskinen skriver

SPEGEPØLSE
RULLEPØLSE
KØDPØLSE
LEVERPØLSE
PØLSE
Unexpected RETURN in 110

TRON sladrer om, hvad der er galt:

. . .
(110)(70) LEVER (80)(100) PØLSE
(110)(100) PØLSE

```
(110)
Unexpected RETURN in 110
```

Efter at programmet egentlig er kørt til ende (med LEVERPØLSE), fortsætter maskinen ned i subrutinen og skriver et ekstra pølse, hvorefter den skal RETURNe. Men vende tilbage hvortil? Børsprogrammet introducerer også et andet nyt reserveret ord, TAB. Prøv

```
1 FOR a=1 TO 20: PRINT TAB(a); a: NEXT
```

så tror jeg, du har fattet galoppen. Lad os nu forsøge os med nogle sunde investeringer:

```
ANTAL AKTIE        KURS

   0    1 Jensen   100
   0    2 Olsen    100
   0    3 Klausen  100
   0    4 Fedesen  100
   0    5 Tomsen   100
   0    6 Borgen   100

KAPITAL: 1000

KØB / SALG (K / S)?
```

Jeg tror, jeg ser tiden an, så jeg taster bare ENTER. Maskinen skriver

```
ANTAL AKTIE        KURS

   0    1 Jensen   95
   0    2 Olsen    101
   0    3 Klausen  90
```

0	4	Fedesen	93
0	5	Tomsen	103
0	6	Borgen	103

KAPITAL : 990

KØB / SALG (K / S) ?

Det lader til, at inflationen udhuler min kapital med 10 kroner pr. omgang, så jeg må hellere købe, mens jeg har noget at købe for. Borgen ser ud til at være solide:

KØB / SALG (K / S) ? K

AKTIENUMMER ? 6

HVOR MANGE ? 5

ANTAL	AKTIE		KURS
0	1	Jensen	97
0	2	Olsen	110
0	3	Klausen	91
0	4	Fedesen	84
0	5	Tomsen	95
5	6	Borgen	113

KAPITAL : 465

KØB / SALG (K / S) ?

Uha! Rygter om en ny roman af Erwin Neutzsky-Wulff:

KØB / SALG (K / S) ? S

AKTIENUMMER ? 6

HVOR MANGE ? 5

Der tjente man såmænd 30 kroner – børsmatador bliver man nok ikke lige med det samme . . . Vi kan så forsøge os som beduin (opgaven går ud på at nå så langt ud i ørkenen som muligt):

```
10 FOR a=1 TO 10:READ a$(a):a(a)=INT(RND*
   1000):NEXT:a=1:b=10:c=10:d=10:e=1
20 CLS:PRINT a;".DAG":PRINT:PRINT "DU HAR
   REJST";f;"KILOMETER":PRINT "DU HAR";
   b;"LITER VAND":IF c>0 THEN PRINT "DU
   ER TIL HEST":PRINT "HESTEN ERm";
   a$(c)
30 IF c=0 THEN PRINT "DU ER TIL FODS"
40 PRINT "DU ERm";a$(d):IF a>10 AND
   g<100 THEN PRINT "RØVERE";g;
   "KILOMETER BORTE"
50 FOR h=1 TO 10:IF ABS(a(h)-f)<10 AND
   c>0 THEN PRINT "HESTEN LUGTER VAND"
60 IF ABS(a(h)-f))<5 THEN PRINT "OASE":
   b=10:d=10:IF c>0 THEN c=10
70 NEXT:IF RND<0.01 THEN PRINT
   "SANDSTORM":b=0:c=c-5:d=d-5
80 PRINT:PRINT "VIL DU":PRINT:PRINT "1.
   REJSE HURTIGERE?":PRINT "2. REJSE
   LANGSOMMERE?":PRINT "3. HVILE
   MEGET?":PRINT "4. DRIKKE?":PRINT "5.
   VENDE OM?":PRINT "6. REJSE SOM FØR?"
90 a$=INKEY$:IF a$="" THEN 90
100 IF ASC(a$)<49 OR ASC(a$)>54 THEN 90
110 ON VAL(a$) GOTO 120,150,190,210,260,
   280
```

```
120 h=h+INT(RND*10): IF c>0 THEN c=c-3
130 IF c=0 THEN d=d-2
140 d=d-1 : GOTO 300
150 h=h-INT(RND*10): IF h<1 THEN h=1
160 IF c>0 THEN c=c-2
170 IF c=0 THEN d=d-1
180 d=d-1 : GOTO 300
190 h=0 : IF c>0 THEN c=c-1
200 d=d-1 : GOTO 300
210 IF b>0 THEN 240
220 IF c>0 THEN c=c-1
230 d=d-1 : GOTO 300
240 b=b-1 : IF c>0 THEN c=c-1
250 d=d+1 : GOTO 300
260 e=e*-1 : IF c>0 THEN c=c-1
270 d=d-1 : GOTO 300
280 IF c>0 THEN c=c-1
290 d=d-1
300 a=a+1 : f=f+(c+d+h)*e : IF f<0 THEN
    PRINT : PRINT "DU ER HJEMME IGEN":
    PRINT : END
310 IF a<11 THEN 350
320 IF f>i THEN i=i+INT(RND*10)
330 IF f<i THEN i=i-INT(RND*10)
340 g=ABS(f-i) : IF g<5 THEN PRINT : PRINT
    "RØVERNE INDHENTEDE DIG" : PRINT : END
350 IF c>10 THEN c=10
360 IF c<0 THEN c=0
370 IF d>10 THEN d=10
380 IF d<=0 THEN PRINT : PRINT "DU ER DØD"
    : PRINT : END
390 GOTO 20
400 DATA DØENDE , MEGET SYG , SYG , UTILPAS ,
    MEGET VARM , TØRSTIG , VARM , VELTILPAS,
    RASK , I TOPFORM
```

Dette program introducerer i øvrigt ON. Prøv

```
10 INPUT a
20 ON a GOTO 30 , 40 , 50
30 PRINT "Du skrev 1" : END
40 PRINT "Du skrev 2" : END
50 PRINT "Du skrev 3"
```

ON a betyder simpelt hen, at a afgør, hvilket af de tre såkaldte destinationstal, maskinen skal bruge. Er a 1, bliver det det første, og maskinen går til 30. Er a 2, bliver det nummer 2, altså 40, og så fremdeles. Vi kan nu prøve vores program:

```
1 . DAG

DU HAR REJST 0 KILOMETER
DU HAR 10 LITER VAND
DU ER TIL HEST
HESTEN ER I TOPFORM
DU ER I TOPFORM

VIL DU

1 . REJSE HURTIGERE ?
2 . REJSE LANGSOMMERE ?
3 . HVILE MEGET ?
4 . DRIKKE ?
5 . VENDE OM ?
6 . REJSE SOM FØR ?

3
```

2 . DAG

DU HAR REJST 18 KILOMETER
DU HAR 10 LITER VAND
DU ER TIL HEST
HESTEN ER RASK
DU ER RASK
HESTEN LUGTER VAND
OASE

1

3 . DAG

DU HAR REJST 45 KILOMETER
DU HAR 10 LITER VAND
DU ER TIL HEST
HESTEN ER VARM
DU ER RASK

4

4 . DAG

DU HAR REJST 72 KILOMETER
DU HAR 9 LITER VAND
DU ER TIL HEST
HESTEN ER TØRSTIG
DU ER I TOPFORM
HESTEN LUGTER VAND

På den 10. dag døde min hest. Jeg var på det tidspunkt nået 126 kilometer ud i ørkenen. 3 dage senere fandt jeg en oase.

Efter yderligere 10 dage i ørkenen overraskedes jeg af en

sandstorm. Jeg havde på dette tidspunkt tilbagelagt ikke mindre end 371 kilometer. Min vandbeholdning var på 3 liter.

24. dag. Ikke mere vand. 378 kilometer ude. Hils fra mig derhjemme . . .

Dette var det sidste, man nogen sinde hørte til AMSTRAD-ekspeditionen sommeren 1985.

Som du vil se, kan man få meget fornøjelse ud af spil af ovenstående type, især hvis man krydrer dem med overraskelser af forskellig art. Ørkenfærden kunne jo eventuelt kombineres med jagten på en forsvunden pyramide.

Mange forbinder vel computer-spil med små mænd, der farer rundt på skærmen. Det kan vi selvfølgelig også lave på AMSTRAD, og det handler næste kapitel om.

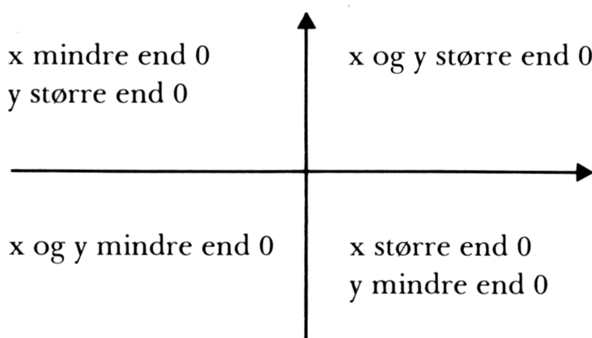
Vi er her ovre i maskinens såkaldte "grafiske" muligheder. Grundlæggende er den *grafiske afbildning*. Tag en tabel som

x	y
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81

Som du ser, har vi i x-kolonnen tallene fra 1 til 9, og i y-kolonnen x * x, altså tallet ganget med sig selv eller i anden potens, tallets kvadrat. Vi kan fortsætte med en sådan tabel ud i det uendelige, eller maskinen kan:

```
10 a=a+1 : PRINT a , a↑2 : GOTO 10
```

Som du vil få et indtryk af, når du kører det, er det ikke særlig overskueligt. Man kan så gribe til grafisk afbildning, hvor man i et såkaldt koordinatssystem afsætter en prik for hvert talpar, idet det første tal, *x-koordinaten*, fortæller, hvor langt *til højre*, prikken skal afsættes, og det andet, *y-koordinaten*, siger, hvor højt oppe. Man kan godt komme både til venstre for og under 0, hvilket giver systemet



Hvordan sætter vi nu en prik på skærmen? Prøv

```
10 CLS : PLOT 10 , 10
```

Prikken kommer til syne helt nede i venstre hjørne, så der er plads til en hel del prikker. Det er altså normalt kun feltet med x og y større end 0, vi får på skærmen. Prøv

```
10 CLS : FOR a=0 TO 399 : PLOT a , a : NEXT
```

Vi får en skrå linje fra nederste venstre hjørne, idet maskinen gennemløber og tegner punkterne (0,0),(1,1),(2,2),(3,3)...,(398,398),(399 , 399). Vi kommer ikke op i højre hjørne. Der kan nemlig være 400 prikker opefter (y-positio-

ner), men – eftersom skærmen er aflang – ikke mindre end 640 x-positioner. Vi kan nu prøve

```
10 CLS : FOR a = 0 TO 639 : PLOT a , a↑2 : NEXT
```

Det giver en meget stejl kurve og

```
OVERFLOW in 10
```

Vi må tilpasse kurven til skærmen, f.eks. med

```
10 CLS : FOR a = 0 TO 639 : PLOT a , a↑2 / 1000 :
   NEXT
```

Vi kan nu "overskue" vores tabel på en helt anden måde. Det er nu, vi kan få brug for vores tal fra fjerde kapitel (stammerne og hjorteflokkene):

```
10 DIM a(50) , b(50) : CLS : FOR a = 1 TO 50 : READ
   a(a) , b(a) : NEXT : FOR a = 1 TO 50 : PLOT a * 12 ,
   a(a) / 5 : NEXT : FOR a = 1 TO 50 : PLOT a * 12 ,
   b(a) / 5 : NEXT
20 GOTO 20
30 DATA 168 , 193 , 64 , 285 , 26 , 488 , 14 , 786 , 10 ,
   1145 , 8 , 1476 , 9 , 1683 , 10 , 1760 , 12 , 1797 , 16 ,
   1801 , 21 , 1799 , 29 , 1783 , 40 , 1752 , 53 , 1739 , 66,
   1705 , 90 , 1659 , 112 , 1605 , 151 , 1537 , 179 , 1442 ,
   231 , 1364 , 268 , 1256 , 320 , 1187 , 360 , 1062 , 421 ,
   960 , 454 , 839
40 DATA 468 , 772 , 477 , 704 , 479 , 662 , 448 , 666 ,
   432 , 646 , 417 , 658 , 397 , 682 , 376 , 727 , 353 , 784,
   348 , 841 , 354 , 889 , 361 , 878 , 377 , 864 , 390 , 845,
   398 , 838 , 410 , 818 , 412 , 789 , 421 , 794 , 413 , 759,
   414 , 775 , 423 , 779 , 415 , 769 , 412 , 750 , 411 , 716,
   401 , 730
```

De sammenhænge, vi før kun anede, fremtræder klart. Endnu tydeligere bliver det, hvis vi erstatter PLOT med DRAW.

```
DRAW x , y
```

betyder: Tegn en ret linje til x,y (fra det sidste punkt). Vi kan altså ikke starte med en DRAW, men må have en PLOT først:

```
10 DIM a(50) , b(50) : CLS : FOR a = 1 TO 50 : READ
   a(a) , b(a) : NEXT : PLOT 12 , a(1) / 5 : FOR
   a = 2 TO 50 : DRAW a * 12 , a(a) / 5 : NEXT : PLOT
   12 , b(1) / 5 : FOR a = 2 TO 50 : DRAW a * 12 , b(a)
   / 5 : NEXT
```

Prøv

```
10 CLS : PLOT 0 , 0 : FOR a = 1 TO 639 : DRAW
   a , SQR(a) : NEXT
```

Her er y-koordinaten altid kvadratroden af x-koordinaten, hvad vi kan udtrykke med ligningen

```
y = SQR(x)
```

grafen og ligningen hører sammen, kurven er ligningens graf, og ligningen kaldes også somme tider for kurvens ligning. Man kan så prøve forskellige ligninger for at se, hvordan de kommer til at se ud, eller omvendt prøve at finde den ligning, der bestemmer en særlig figur. Vi kan få en såkaldt sinuskurve (SINus er endnu en funktion) med

```
10 CLS : FOR a = 0 TO 8 STEP 0.025 : PLOT a * 80,
   SIN(a) * 200 + 199 : NEXT
```

Prøv

```
10 CLS : PLOT 320 , 399 : FOR a = 0 TO 2 STEP
   0.05 : DRAW SIN(PI * a) * 200 + 320 , COS
   (PI * a) * 200 + 199 : NEXT
```

eller

```
10 MODE 2 : FOR a = 5 TO 200 STEP 5 : PLOT
   320 , a + 199 : FOR b = 0 TO 2 STEP 0.05 :
   DRAW SIN(PI * b) * a + 320 , COS(PI * b) * a + 199 :
   NEXT b , a
```

eller

```
10 CLS : FOR a = 0 TO 2 STEP 0.01 : PLOT 320 ,
   199 : DRAW SIN(PI * a) * 200 + 320 , COS(PI * a) * 200
   + 199 : NEXT
```

eller

```
10 MODE 2 : PLOT 320 , 199 : FOR a = 0 TO 10
   STEP 0.1 : FOR b = a TO a + 2 STEP 0.5 :
   DRAW SIN(PI * b) * a * 25 + 320 , COS(PI * b) * a * 25 +
   199 : NEXT b , a
```

Du kan også simpelt hen flytte til et andet punkt:

```
10 CLS : FOR a = 0 TO 390 STEP 10 : MOVE 0 , a :
   DRAW 639 , 0 : NEXT
```

Alle disse reserverede ord, PLOT, DRAW og MOVE kan i øvrigt blive til tre nye: PLOTR, DRAWR og MOVER. R står for Relativt. En oversigt:

PLOT x , y betyder sæt en prik i x , y
DRAW x , y betyder træk en streg til x , y
MOVE x , y betyder flyt til x , y
PLOTR x , y betyder sæt en prik x til højre og y over nuværende position
DRAWR betyder træk en streg x til højre, y op
MOVER x , y betyder flyt x til højre, y op

Lad os tage et par eksempler:

```
10 CLS : PLOT  100 , 200 : PLOT  200 , 200 : PLOT
   200 , 100 : PLOT  100 , 100
```

giver fire prikker.

```
10 CLS : PLOT  100 , 200 : DRAW  200 , 200 : DRAW
   200 , 100 : DRAW 100 , 100 : DRAW  100 , 200
```

vil forbinde disse punkter til et kvadrat. Dette kvadrat kan også tegnes med

```
10 CLS : PLOT  100 , 200 : DRAW  100 , 0 : DRAWR
   0 , – 100 : DRAWR  – 100 , 0 : DRAWR  0 , 100
```

Vi kan i øvrigt få opgivet den ”grafiske markørs” øjeblikkelige position med henholdsvis

XPOS

og

YPOS

```
 10 MODE 0 : PLOT 320 , 200
 20 a$= INKEY$: a$= UPPER$(a$) : IF a$=”W”
    THEN DRAWR 0 , 1
 30 IF a$= ”Z” THEN DRAWR 0 , - 1
 40 IF a$= ”A” THEN DRAWR - 1 , 0
 50 IF a$= ”S” THEN DRAWR 1 , 0
 60 IF XPOS = 0 THEN MOVER 1 , 0
 70 IF XPOS = 640 THEN MOVER - 1 , 0
 80 IF YPOS = 0 THEN MOVER 0 , 1
 90 IF YPOS = 400 THEN MOVER 0 , - 1
100 GOTO 20
```

Dette program er egentlig på 13 linjer. Linje 20 er således f.eks. sammensat af 5 linjer.

Denne sammenpresning foretager vi givetvis af hensyn til den anvendte hukommelse, men dog i langt højere grad på grund af den større overskuelighed. Vi må kunne overse et program for at gøre det så godt som muligt.

Idealet bliver det kompakte, hurtige, effektive, overskuelige og letforståelige program. Jo mere vi kan presse linjer, der hjælper hinanden med at udføre bestemte opgaver, sammen til faste, omflyttelige ”klodser”, jo mere kan vi koncentrere os om de store linjer i vores program.

Det synes måske krukket i programmer af denne størrelse, men forestil dig et program af tidobbelt længde. Så er det rart, hvis vi kan finde startstillingen og ikke løber vild i et virvar af overflødige GOTOer og linjenumre.

Betragter vi nu vores program med disse øjne, må det slå os, at der er noget påfaldende ”uforkortet” ved linjerne 30-90. Men hvad skal man gøre, det er IF-linjer, som man ikke uden videre kan hægte sammen. Her kommer imidlertid et andet fænomen os til hjælp: SANDHEDSVÆRDIEN og DET BETINGEDE UDTRYK. Hvad betyder egentlig

IF a = 1 THEN 666

Jo, maskinen skal gå til linje 666, hvis og kun hvis det er opfyldt eller SANDT, at a = 1. Men hvordan ved maskinen nu det? Jo, den tilkender nemlig helt automatisk enhver relation et tal, afhængigt af, om relationen er sand eller falsk, en såkaldt SANDHEDSVÆRDI, nemlig

– 1 (minus 1) hvis relationen er sand

og

0 (nul) hvis relationen er falsk

Det kan du selv kontrollere. Skriv

? "2 + 2" (skriv strengen "2 + 2")

derefter

? 2 + 2 (skriv resultatet af 2 + 2)

og endelig

? 2 + 2 = 4 (skriv SANDHEDSVÆRDIEN af relationen 2 + 2 = 4)

Er du med? Prøv nu

? 2 + 2 = 5

Eller

? 2 + 2 = 4 AND 2 + 2 = 5

Forklar det sidste resultat. Lad os nu vende tilbage til vores program. Tag f.eks.

```
IF A$="Z" THEN DRAWR 0,-1
```

Hvad, om vi i stedet skrev

```
DRAWR 0,a$="Z"
```

Ja, hvis a$="Z", så er sandhedsværdien af relationen – 1, og vi får

```
DRAWR 0,-1
```

Hvis derimod a$ ikke er "Z", bliver sandhedsværdien 0, og vi får

```
DRAWR 0,0
```

Pengene passer. Og dermed er de 13 linjer, der blev til 10, faktisk blevet til TO! Nemlig

```
10 MODE 0:PLOT 320,200
20 a$=INKEY$:a$=UPPER$(a$):DRAWR a$="A",
   a$="Z":DRAWR -(a$="S"),-(a$="W"):
   MOVER XPOS=640,YPOS=400:MOVER
   -(XPOS=0),-(YPOS=0):GOTO 20
```

Men sæt nu, vi vil bruge markørpilene i stedet for AZSW? Vi kan jo ikke skrive a$="markørpil op". Derfor har INKEY$ også en lillebror

```
INKEY
```

Prøv

```
10 CLS
20 LOCATE 1,1:PRINT INKEY(79);"m":GOTO 20
```

Maskinen skriver – 1. Prøv nu at taste DEL (tasten). Nu skriver den 0. Tast DEL sammen med SHIFT. Sammen med CTRL. Sammen med SHIFT og CTRL. Du får værdierne 32, 128 og 160.

```
IF INKEY(79)=32
```

ville altså betyde: Hvis DEL og SHIFT er holdt nede. 79 står for DEL. Tilsvarende har du

ESC 66
TAB 68
CAPS LOCK 70
SHIFT 21
MELLEMRUM 47
CLR 16
STOR ENTER 18
CTRL 23
LILLE ENTER 6
COPY 9
MARKØR OP 0
MARKØR NED 2
MARKØR TIL VENSTRE 8
MARKØR TIL HØJRE 1

hvilket giver os

```
10 MODE 0:PLOT 320,200
20 DRAWR NOT INKEY(8),NOT INKEY(2):DRAWR
```

```
-NOT INKEY(1),-NOT INKEY(0):MOVER
XPOS=640 , YPOS=400 : MOVER -(XPOS=0),
-(YPOS=0):GOTO 20
```

7. INVASION FRA MARS

FRE – BIN$ – HEX$ – PEN – PAPER – INK – BORDER – KEY – SPEED INK

I skolen lærte vi at regne med tal på mere end et ciffer ved f.eks. at opsplitte 1234 i

1 tusind
2 hundrede
3 tiere
4 enere

Vores "almindelige" talsystem er et titalssystem, dvs. det består af 10 tal: 0, 1, 2, 3, 4, 5, 6, 7, 8 og 9. Når vi har talt op til 9, får vi 0 igen på ener-pladsen og begynder at tælle på tier-pladsen med 1, op til 9 tiere og 9 enere, hvor vi må tage hundrede-pladsen i brug.

Vi kan imidlertid også lave et talsystem med kun to tal, 0 og 1. Nu skal vi skifte plads allerede, når der står 1 på denne plads, det højest tilladelige.

Vi skifter altså første gang plads ved 2, der bliver 10. Derefter bliver 3 til 11, og 4 bliver til 100. Vi har altså

TOTALSSYSTEM	TITALSSYSTEM
0	0
1	1
10	2
11	3
100	4
101	5
110	6

111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15
10000	16

Upraktisk? Ikke for en maskine. Nu kan nemlig ethvert tal udtrykkes ved en serie elementer, der enten er tændte eller slukkede. Tallet 13 kræver fire lamper, og 1101 udtrykkes

TÆNDT TÆNDT SLUKKET TÆNDT

altså tændt for 1 og slukket for 0. Totalssystemet hedder på engelsk BINARY SYSTEM, og derfor kalder man et element, der ved at være tændt eller slukket udtrykker et ciffer i totalssystemet, BINARY DIGIT (digit = ciffer) eller simpelt hen BIT. For at få "rigtige" tal er disse bits ordnet i enheder på 8 bits, og disse enheder kaldes en BYTE.

En byte kan (regn efter) udtrykke et tal mellem 0 og 255. Der er 256 ASCII-koder, hvilket vil sige, at en hvilken som helst af ASCII-systemets karakterer kan udtrykkes ved et tal mellem 0 og 255 og altså af 1 byte.

Læg mærke til, at det ikke er ligegyldigt, hvilke bits der er tændt i en byte. 10110010 er noget andet end 11001001.

På samme måde skal flere bytes naturligvis også arbejde sammen, når hele BASIC-sprogets utallige muligheder skal udtrykkes. Hver byte har sin plads eller ADRESSE. MASKINE er jo heller ikke det samme som ANEMISK.

Bytes tælles i K. 1 K eller kilo-byte er 1024 bytes.

64K skulle altså give dig godt og vel 64000 bytes at arbej-

de med. I virkeligheden bruger maskinen selv nogle af dem. Du kan altid få den tilrådighedstående hukommelse med

```
? FRE(0)
```

Tallet i parentesen er ligegyldigt. Når du netop har tændt maskinen, er antallet af frie bytes 43533 eller cirka 42½ K.

Alt dette er RAM eller Random Access Memory, dvs. hukommelse, som du kan bruge, som du vil, til programmer, variabler, etc. Men maskinen indeholder naturligvis også en for-programmeret viden om, hvordan man lægger sammen, uddrager kvadratrødder, eller blot oversætter dit halv-engelske BASIC til "binærkode" eller "maskinkode".

RAM-adresserne er opdelt i større "byer", hver med deres egen funktion. Der er særlige områder for oplagring af programmer, variabler, skærmbilledet, etc.

Den "egentlige" maskine er den såkaldte CPU, "Central Processing Unit". Det er "regneværket" i AMSTRAD, og det står i forbindelse med alle de forskellige former for hukommelse.

Prøv

```
10 INPUT a$: a=LEN(a$): FOR b=1 TO a: c=c+
   VAL(MID$(a$, b, 1))* 2↑(a-b): NEXT: PRINT c:
   RUN

RUN
? 1101
 13
? 101010101
 341
?
```

og

```
10 INPUT a
20 b=b+1 : IF a>=2↑b THEN 20
30 FOR c=b-1 TO 0 STEP -1 : IF a>=2↑c
   THEN PRINT "1"; : a=a-2↑c : GOTO 50
40 PRINT "0";
50 NEXT : PRINT : RUN

RUN
? 12
1100
? 1000
1111101000
?
```

Vi kan checke begge programmer ved at sætte dem sammen til et:

```
10 INPUT "BIN"; a$: a=LEN(a$): FOR b=1 TO a:
   c=c+VAL(MID$(a$, b, 1))*2↑(a-b): NEXT :
   PRINT c : CLEAR : INPUT "DEC"; a
20 b=b+1 : IF a>=2↑b THEN 20
30 FOR c=b-1 TO 0 STEP -1 : IF a>=2↑c
   THEN PRINT "1"; : a=a-2↑c : GOTO 50
40 PRINT "0";
50 NEXT : PRINT : RUN

BIN? 1001001001
 585
DEC ? 585
1001001001
BIN ? 10
 2
DEC ? 2
10
```

```
BIN ? 10101010101010101010
 699050
DEC ? 699050
10101010101010101010
```

Det ser ud til at virke. Maskinen kan klare noget af det med egne funktioner. Prøv

```
? BIN$(585)
1001001001
```

Den kan den nok klare, men ikke

```
? BIN$(699050)
```

Den kan højst omsætte 65535 til binært tal. Den anden vej kommer vi med

```
? &X1001001001
```

Maskinen kan også regne heksadecimalt, efter 16-tals-systemet:

SEKSTENTALSSYSTEM	TITALSSYSTEM
1	1
2	2
3	3
......	......
8	8
9	9
A	10
B	11
C	12
D	13

E	14
F	15
10	16
11	17
12	18
.....	.....
18	24
19	25
1A	26
1B	27
1C	28

Læg mærke til, at i totalssystemet får vi cifre fra vores eget talsystem til overs, i 16-talssystem løber vi tværtimod tør og må bruge bogstaverne A–F for de 6 manglende cifre. 2A bliver altså $2*16+10=42$, og $AA=10*16+10=170$. Prøv

```
? HEX$(1000)
3E8
```

og

```
? &3E8
1000
```

ikke at forveksle med

```
? 3E8
 300000000
```

Vi kan nu forstå begrebet skærmhukommelse. Skriv

```
MODE 0
```

Skærmen består nu af 25 linjer med 20 pladser i hver. Maskinen må naturligvis vide, hvad der står på hver af disse 500 pladser for at vise det på skærmen.

Derfor har hver plads også fået en byte med en adresse. Som vi har set, kan hver byte indeholde et tal op til 255. 65 i en skærmadresse betyder et stort A på den pågældende plads.

Hvis AMSTRAD var en sort-hvid datamat, ville dette være nok. Nu har hver plads imidlertid fået tildelt en ekstra byte, der fortæller maskinen noget om dets farve.

Hvert felt har jo nemlig to farver: Karakterens farve og baggrundens farve. Det bliver lige en halv byte til hver.

4 bits kan udtrykke tallene 0–15. Vi kan altså vælge imellem 16 karakterfarver og 16 baggrundsfarver.

Mange datamater har kun 16 farver i sig. AMSTRAD har 27!

Men hvad nytter det, når maskinen kun "forstår" 16? De sidste 10 bliver jo aldrig brugt!

For at løse dette problem har AMSTRAD opfundet sin egen metode, der kan virke forvirrende i begyndelsen. Vi skelner mellem på den ene side PEN og PAPER, og på den anden side INK. PEN og PAPER er ikke umiddelbart noget problem.

```
PEN 7
```

betyder brug farvepen nummer 7, og tilsvarende

```
PAPER 9
```

på farvepapir nummer 9. Men hvad er så INK? Vi kan sammenligne INK med en palet. På en palet kan der måske være seksten farver. Det betyder ikke, at maleren ikke har flere i sin kasse. Det betyder bare, at han, inden han be-

gyndte at male, udvalgte de seksten, han skulle bruge til netop dette maleri, og kom på paletten.

I rum nummer 1 (på paletten) kom han måske rød. En anden dag er der måske blåt i dette rum.

AMSTRAD betjener sig af et lignende system. Derfor betyder PEN 1 måske somme tider rød og til andre blå.

Om det skal betyde det ene eller det andet, bestemmer vi, når vi udvælger farver fra malerkassen til at komme på paletten. Det er her, INK kommer ind.

INK 1 , 7

betyder: Kom farve 7 i det første rum på paletten. Nummer 7 i malerkassen er i øvrigt violet, så her er altså violets plads på paletten. Det vil igen sige, at

PEN 1

fra nu af betyder: Brug farven i rum 1 på paletten, det vil sige violet. Lad os se, hvordan det virker. Her er for det første farverne med deres numre i malerkassen:

0 SORT
1 BLÅ
2 KRAFTIGT BLÅ
3 RØD
4 MAGENTA
5 LILLA
6 KRAFTIGT RØD
7 VIOLET
8 KRAFTIGT MAGENTA
9 GRØN
10 CYAN
11 HIMMELBLÅ
12 GUL

13 HVID
14 PASTELBLÅ
15 ORANGE
16 LYSERØD
17 PASTEL MAGENTA
18 KRAFTIGT GRØN
19 SØGRØN
20 KRAFTIGT CYAN
21 LIME
22 PASTELGRØN
23 PASTELCYAN
24 KRAFTIGT GUL
25 PASTELGUL
26 KRAFTIGT HVID

Vi kan få alle disse farver at se på en forholdsvis enkel måde. Kanten omkring skærmen kan nemlig også bringes til at skifte farve. Her vil

```
BORDER 3
```

temmelig ukompliceret give en rød skærmkant. Prøv

```
10 CLS : FOR a = 0 TO 26 : BORDER a : LOCATE 1,
   1 : PRINT "KANTFARVE" ; a : FOR b = 1 TO
   1000 : NEXT b , a
```

Prøv nu

```
INK 2 , 26
```

Vi har kommet kraftigt hvidt i rum nummer 2 på vores palet. Hvis vi nu stikker vores pensel i dette rum

```
PEN 2
```

begynder vi at skrive hvidt, vi kan allerede se virkningen på vores Ready.

Hvad skete der? Jo, maskinen kiggede efter i vores halve byte og fandt tallet 2. Derefter løb den et andet sted hen og så efter, hvad der var i dette rum på paletten. Skift tilbage til

```
MODE 1
```

Nu har vi dobbelt så mange pladser pr. linje eller 1000 pladser at fordele på 500 halve bytes. Det giver 2 bits til hver.

Nu kan du altså kun male med 4 farver, det vil sige, du kan kun fylde 4 rum på paletten, inden du begynder at male. Dette viser sig i praksis ved, at maskinen vil opfatte

```
PEN 10
```

som

```
PEN (10 - 8) = PEN 2
```

Dette svarer til, at den ikke læser de to første af de fire bits, der udtrykker tallet 0–15. I MODE 2 ryger der endnu en bit (80 pladser pr. linje),

```
PEN 10
```

bliver til

PEN ~~101~~0

altså

```
PEN 0
```

mens f.eks.

PEN 7

bliver til

PEN Ø111

eller

PEN 1

Forvirret? Vel, pointen er, at i MODE 0 kan du tilskrive INK 0–15 (og altså bruge PEN og PAPER 0-15 med forskelligt resultat), i MODE 1 INK 0–3, og endelig i MODE 2 INK 0–1. Vi kan også udtrykke det, som at du afhængig af MODE kan have 16, 4 eller 2 farver på skærmen på samme tid.

Som du vil forstå, kan man meget let ende et sted med samme farve PEN og PAPER og så mange INK-omflytninger, at man ikke aner, hvor man er, eller kan læse sig frem til det på skærmen. Vi kunne altså godt bruge en-"vend-til-bage-til-normaltilstanden"-knap.

Du kan altid taste CTRL-SHIFT-ESC, men det er det samme som at slukke for maskinen, og måske gemmer der sig et værdifuldt program derinde. Vi må altså til at definere en ny tast. Det kan vi gøre med

KEY

Først skal vi finde en tast, vi kan undvære. Hvad med regnemaskinetastaturet til højre? Disse taster har numrene (med henblik på definering af taster):

0 128
1 129
2 130
3 131
4 132
5 133
6 134
7 135
8 136
9 137
. 138
ENTER 139

Hvis vi nu f.eks. vælger 138, kan vi have

KEY 138 , "AMSTRAD"

Nu skriver tasten ikke længere et punktum, men hele ordet "AMSTRAD". Prøv nu

KEY 138 , "EDITm"

Nu kan du, hvis du skal redigere en linje blot taste . + linje-nummer + ENTER. Du kan tilsvarende have

KEY 138 , "LIST"

men vi kan gå endnu længere. Prøv

KEY 138 , "LIST" + CHR$(13)

ENTER havde jo ASCII 13, ikke? Nu lister maskinen i det øjeblik, du taster punktummet. Men det var farverne, vi kom fra:

```
KEY 138,"INK 0,1:INK 1,24:PAPER 0:PEN 1"
+CHR$(13)
```

Prøv nu

```
INK 0,6
```

og

```
INK 1,7
```

Ikke just den mest læselige kombination. Tast da blot det forløsende punktum, og du kan igen se, hvad der foregår. Du kan nu trygt gå i gang med at lege med farverne. Prøv i MODE 1

```
10 CLS:a$=CHR$(143):BORDER 26:INK 0,26:
   FOR a=1 TO 3:INK a,INT(RND*26):NEXT:
   PAPER 0
20 PEN INT(RND*3)+1:a=INT(RND*13):b=
   INT(RND*13):LOCATE 20-a,13-b:PRINT a$
   :LOCATE 20-a,13+b:PRINT a$:LOCATE
   20+a,13-b:PRINT a$:LOCATE 20+a,13+b
   :PRINT a$:LOCATE 20-b,13-a:PRINT a$:
   LOCATE 20-b,13+a:PRINT a$:LOCATE
   20+b,13-a:PRINT a$:LOCATE 20+b,13+a
   :PRINT a$
30 IF RND<0.001 THEN PRINT CHR$(7);:FOR
   a=1 TO 10000:NEXT:RUN
40 GOTO 20
```

eller

```
10 CLS:BORDER 26:INK 0,26:FOR a=1 TO 3:
   INK a,INT(RND*26):NEXT:PAPER 0
20 c=INT(RND*3)+1:a=INT(RND*200):b=INT
```

```
   (RND*200): PLOT 320-a, 200-b, c: PLOT
   320-a, 200+b, c: PLOT 320+a, 200-b, c:
   PLOT 320+a, 200+b, c: PLOT 320-b, 200-a,
   c: PLOT 320-b, 200+a, c: PLOT 320+b, 200-
   a, c: PLOT 320+b, 200+a, c
40 GOTO 20
```

Læg mærke til, at

```
PEN c: LOCATE a, b: PRINT a$
```

her udskiftes med

```
PLOT a, b, c
```

Den PEN, du vælger, bliver en slags ”tredje koordinat”. Prøv at udskifte PLOT med DRAW. Du kan også blinke mellem to farver, prøv

```
BORDER 6, 26
```

eller

```
INK 1, 6, 26
```

fulgt af

```
PEN 1
```

Du kan endda bestemme, hvor hurtigt den skal blinke med

```
SPEED INK
```

Prøv

```
10 CLS : BORDER 0 : INK 0 , 0 , 26 : PAPER 0 :
   SPEED INK 2 , 1
20 GOTO 20
```

Når du har siddet og gloet lidt på det, vil du opdage, at det hvide kvadrat trækker sig tilbage; det er lidt ligesom en tunnelbane. Så skal du enten trække øjnene til dig eller have fat i en anden bog af samme forfatter betitlet OKKULTISME.

Det første tal i SPEED INK-kommandoen fortæller, hvor længe den første farve (i dette tilfælde 0 = sort) står på skærmen målt i halvtredsindstyvendele af et sekund. Det andet angiver det samme for nummer tos vedkommende. Skriv

```
10 CLS : a$= CHR$(231) : BORDER 0 : INK 0 , 26 :
   INK 1 , 6 : PAPER 0 : PEN 1 : LOCATE 10 , 10
   : PRINT a$
20 GOTO 20
```

Lad os se, om vi kan få lidt fart på den lille bold.

```
10 CLS : a$= CHR$(231) : a = 10 : b = 10 : BORDER 0 :
   INK 0 , 26 : INK 1 , 6 : PAPER 0 : PEN 1
20 LOCATE a , b : PRINT "m" : a = a + 1 : IF a = 41
   THEN a = 1
30 LOCATE a , b : PRINT a$: FOR p = 1 TO 50 :
   NEXT : GOTO 20
```

Kan du se, hvad der sker? a er boldens plads, og b dens linje. De er begge to som før til at begynde med 10.

I linje 20 sker der noget. Det er ikke bolden, der bliver skrevet, men et mellemrum.

Dette skyldes, at bolden jo skal flyttes, det vil sige, den "gamle" bold skal væk. Når dette er sket, flytter vi en plads

til højre (a = a + 1). Hvis a derved bliver 41, bliver den i stedet til 1, så vi ikke kommer ud over kanten: Bolden begynder forfra igen fra venstre.

I linje 30 kommer så endelig bolden. Derpå følger en pause, så vi når at se den, og det hele begynder forfra igen. Ville det nu ikke være smart, hvis den røde bold ligesom en billardkugle, når den stødte mod den sorte bande, løb den anden vej? Så skal vi bruge et flag:

```
10 CLS : a$= CHR$(231) : a = 10 : b = 10 : c = 1 :
   BORDER 0 : INK 0 , 26 : INK 1 , 6 : PAPER 0 : PEN
   1
20 LOCATE a , b : PRINT "m" : a = a + c : LOCATE
   a , b : PRINT a$: IF a = 1 OR a = 40 THEN
   c = - c
30 FOR p = 1 TO 50 : NEXT : GOTO 20
```

Det er c, der er "flaget". Det er også det, der lægges til a.

Når a bliver 0 eller 40, bliver det ganget med – 1: Hvis det var 1, bliver det – 1, og bolden ruller til venstre (a + c bliver a + (– 1) = a – 1. Hvis det var – 1, bliver det atter –(– 1) = 1, og bolden ruller atter til højre. To flag ville give

```
10 CLS : a$=CHR$(231) : a = 10 : b = 10 : c = 1 : d = 1 :
   BORDER 0 : INK 0 , 26 : INK 1 , 6 : PAPER 0 :
   PEN 1
20 LOCATE a , b : PRINT "m" : a = a + c : b = b + d :
   LOCATE a , b : PRINT a$: IF a = 1 OR a = 40
   THEN c = - c
30 IF b = 1 OR b = 25 THEN d = - d
40 FOR p = 1 TO 50 : NEXT : GOTO 20
```

Det er jo næsten tennis, det her, men så skal vi også lige have et bat:

```
10 CLS : a = 1 : b = 1 : c = 1 : d = 1 : e = 20 : BORDER 9
   : INK 0 , 25 : INK 1 , 6 : INK 2 , 1 : PAPER 0
20 LOCATE a , b : PRINT "m" : a = a + c : b = b + d :
   LOCATE a , b : PEN 1 : PRINT CHR$(231) : IF
   a = 1 OR a = 40 THEN c = - c
30 IF b = 1 OR b = 24 THEN d = - d
40 LOCATE e , 25 : PRINT "mmm" ; : e = e - NOT
   INKEY(1) + NOT INKEY(8) : e = e + (e = 39) - (e = 0) :
   LOCATE e , 25 : PEN 2 : PRINT CHR$(208) ; CHR$
   (208) ; CHR$(208) ; : IF b = 24 THEN f = f + 1 :
   PRINT CHR$(7) ; : IF (a<e OR a>e + 2) THEN
   60
50 FOR g = 1 TO 10 : NEXT : GOTO 20
60 LOCATE 1 , 1 : PRINT "Du ramte bolden" ; f - 1 ;
   "gange"
70 GOTO 70
```

eller

10 Slet skærm. Startværdier. Grøn kant. Lysegult, grønt og blåt "blæk". Papir lysegult.
20 Slet bold. Ny plads. Skriv bold med rødt. Bold for langt til venstre eller højre?
30 Bold for langt oppe eller nede?
40 Slet bat. Ny plads fra tastatur. Bat for langt til venstre eller højre? Skriv bat med blåt. Rammer battet? I så fald points.
50 Pause. Forfra igen.
60 Slut. Hvor mange points?
70 "Frysning" af billede.

Der er også temmelig mange variabler at holde rede på:

a boldens plads
b boldens linje

c boldens pladsflag
d boldens linjeflag
e battets plads
f points
g kontrolvariabel

Prøv

```
 10 MODE 0 : a = 3 : b = 3 : c = 1 : d = 1 : e = 1 : g = 1 :
    h = 1 : i = 3 : j = 3 :BORDER 19 : INK 0 , 26 : INK
    1 , 0 : INK 2 , 1 : INK 3 , 6 : INK 4 , 9 : INK 5 , 7
    : INK 6 , 5 : PAPER 0
 20 LOCATE a , b : PRINT "m" : a = a + c : b = b + d
    : LOCATE a , b : PEN 1 : PRINT CHR$(226) : IF
    a = 1 OR a = 20 OR RND < 0.05 THEN c = - c
 30 IF b = 2 OR b = 20 OR RND < 0.05 THEN d = - d
 40 LOCATE e , 25 : PRINT "m" ; : e = e - NOT
    INKEY(1) + NOT INKEY(8) : e = e + (e = 21) - (e = 0) :
    LOCATE e , 25 : PEN 2 : PRINT CHR$(244) ; : IF
    NOT INKEY(47) AND f = 0 THEN f = 1 : g = e :
    h = 25
 50 IF f = 0 THEN 80
 60 LOCATE g , h : PRINT "m" ; : h = h - 1 : IF h = 1
    THEN f = 0 : GOTO 80
 70 LOCATE g , h : PEN 3 : PRINT CHR$(239) : IF
    a = g AND b = h THEN PEN 5 : FOR p = 1 TO
    10 : LOCATE a , b : PRINT CHR$(238) : FOR
    q = 1 TO 10 : LOCATE a , b : PRINT "m" :NEXT
    q , p : m = m + 1 : LOCATE 1 , 1 : PEN 6 : PRINT
    "POINTS" ; m : a = 3 : b = 3 : f = 0
 80 LOCATE i , j : PRINT " " ; : j = j + 1 : IF j = 26
    THEN i = a : j = b
 90 LOCATE i , j : PEN 4 : PRINT CHR$(252) ;
100 IF i = e AND j = 25 THEN 120
```

```
110 FOR p=1 TO 10 : NEXT : GOTO 20
120 SPEED INK 2 , 2 : INK 0 , 0 , 3 : FOR p=1 TO
    1000 : NEXT : INK 0 , 9
130 GOTO 130
```

FARVER

BORDER 19 (søgrøn)
INK 0 (papir) 26 (kraftigt hvid)
INK 1 0 (sort)
INK 2 1 (blå)
INK 3 6 (kraftigt rød)
INK 4 9 (grøn)
INK 5 7 (violet)
INK 6 5 (lilla)

KARAKTERER

226 rumskib – egentlig klør (sort)
244 base (blå)
239 raket (kraftigt rød)
252 bombe (grøn)
238 eksplosion (violet)

VARIABLER

a rumskibets plads
b rumskibets linje
c rumskibets pladsflag
d rumskibets linjeflag
e basens plads
f raketflag
g rakettens pladsflag
h rakettens linjeflag
i bombens pladsflag

j bombens linjeflag
m points
p kontrolvariabel
q kontrolvariabel

BESKRIVELSE

10 MODE 0. Startværdier. Søgrøn kant. Farver. Kraftigt hvidt papir.

20 Slet rumskib. Flyt rumskib. Skriv rumskib med sort. For langt til venstre eller højre? Læg mærke til tilfældighedsfaktoren, som gør rumskibets kurs mere uforudsigelig og vanskeligere at ramme.

30 Rumskib for langt oppe eller nede? Tilfældighedsfaktor.

40 Slet base. Aflæs tastatur (markørtaster – raket affyres med mellemrumstasten). Base for langt til venstre eller højre? Skriv base med blåt. Hvis mellemrumstasten trykket ned og raket til rådighed, affyr raket. Raketflag=1 viser raket undervejs. g og h placerer raket med udgangsposition i basens koordinater.

50 Hvis raketflag=0 (ingen raket undervejs), spring raketrutine 60-70 over.

60 Slet raket. Flyt raket, 1 lige op. Hvis raket forsvinder over overkant, ny raket til rådighed, raketflag=0. Ud af raketrutine.

70 Skriv raket med kraftigt rødt. Hvis raket truffet rumskib, eksplosion, pointsforøgelse. Skriv nyt pointsantal øverst til venstre på skærmen med lilla. Nyt rumskib. Ny raket til rådighed.

80 Slet bombe. Flyt bombe, 1 lige ned. Hvis bombe rammer jorden, fæld ny bombe med udgangsposition i rumskibets koordinater.

90 Skriv bombe med grønt.

100 Hvis bombe ramt base, ud af løkke.
110 Pause. Om igen.
120 Sluteffekt, skærm blinker sort/kraftigt rødt og bliver endelig grøn.
130 Frysning.

Så simpelt dette spil ellers er, indeholder det, hvis du kigger efter, svar på grundlæggende spørgsmål om fremstilling af "TV-spil", spørgsmål som

Hvordan lægger jeg farver på?

Hvordan får jeg ting til at bevæge sig?

Hvordan forhindrer jeg dem i at ryge ud over kanten, så jeg får IMPROPER ARGUMENT og andre ubehagelige fejlmeddelelser, som standser mit spil?

Hvordan flytter jeg ting på skærmen med tastaturet?

Hvordan skyder jeg på rumskibene og skyder dem ned?

Hvordan får jeg dem til at skyde på mig?

Hvordan holder jeg rede på det hele?

Studér programmet grundigt ved hjælp af beskrivelsen, så vil du hurtigt lære principperne og kunne lave langt mere fantasifulde programmer. Her er det gjort så enkelt og overskueligt som muligt.

I næste kapitel skal vi gå et skridt videre i vores spilmageri. Så vil vi ikke længere være tilfredse med det indbyggede udvalg af karakterer.

Vi skal med andre ord til at lave vores egne rumskibe og uhyrer, ja, måske et helt nyt alfabet. Vi kunne f.eks. prøve

at få maskinen til at skrive Æ og Å, ligesom et lille ø heller ikke ville være af vejen (for det store bruger vi jo nul).

Så er det der, hvor vores spil begynder for alvor at se ud af noget. Vi er kommet langt fra vores første regnestykker og variabler.

En anden alvorlig mangel ved vores hidtidige spil har været lyden. Det er også noget, vi skal udforske.

I øvrigt er dette måske stedet til en lille repetition. Vær lige sikker på, at du har grundbegreberne i orden. Det begynder så småt at blive indviklet, det her . . .

8. ANDRE FUNKTIONER

SYMBOL – SYMBOL AFTER – INSTR – WHILE – WEND – AFTER – REMAIN – EVERY – REM – AUTO – WINDOW – PEEK – POKE – MAX – MIN – SPACES$ – ZONE – DEF FN

I det forrige kapitel lavede vi et spil, hvor vi skød rumskibe ned. Egentlig var det ikke noget rigtigt rumskib, men kortfarven klør. Det var det sådan set nødt til at være, eftersom rumskibe ikke fandtes i karaktersættet.

Vi kan imidlertid også definere vore egne karakterer. Så skal vi bare vide lidt om, hvordan maskinen laver sine. Tag et bogstav som A. Som alt andet på skærmen er det sammensat af små prikker, og hvis vi forstørrede det op, ville vi se noget i retning af

```
   **
  ****
 **  **
 **  **
 ******
 **  **
 **  **
```

I virkeligheden er dette oplagret i maskinen som

```
00011000
00111100
01100110
01100110
01111110
01100110
01100110
00000000
```

eller de 8 bytes

```
 24
 60
102
102
126
102
102
  0
```

Skal vi lave vores egne karakterer, må vi for hver karakter, vi definerer, made 8 tal ind i maskinen. Vi kunne f.eks. prøve at lave et Æ:

```
 ******
** ** *
** **
******
** **
** ** *
** ****
```

Læg mærke til, at karakteren kun er 7 * 7. Det sørger for mellemrum til de andre bogstaver. Tallene bliver da

```
126
218
216
252
216
218
222
  0
```

Prøv nu

```
10 SYMBOL 255 , 126 , 218 , 216 , 252 , 216 , 218 , 222,
   0 : PRINT CHR$(255)
```

Læg mærke til, at der nu er 9 tal. Det første (255) betyder blot: Skift karakteren med ASCII 255 ud med følgende karakter. De sidste 16 karakterer (240–255) er umiddelbart definerbare på denne måde. Dette antal kan vi imidlertid ændre med

```
SYMBOL AFTER
```

Således betyder

```
SYMBOL AFTER 230
```

at grænsen går ved 230 i stedet for 240, hvilket giver os 26 definerbare karakterer.

Nu er det jo lidt upraktisk at skulle skrive

```
PRINT "K"; CHR$(255); "RLIGHED"
```

Vi skulle helst kunne få bogstavet direkte fra tastaturet. Den er jo selvfølgelig imidlertid let klaret med

```
10 SYMBOL 255 , 126 , 218 , 216 , 252 , 216 , 218 , 222,
   0 : KEY 128 , CHR$(255)
```

Nu kan du bruge 0 på højretastaturet som Æ. Du kan selvfølgelig også få et lille æ:

```
10 SYMBOL 255 , 126 , 218 , 216 , 252 , 216 , 218 , 222,
   0 : KEY 128 , CHR$(255)
```

```
20 SYMBOL 254 , 0 , 0 , 252 , 54 , 254 , 176 , 252 , 0 :
   KEY 129 , CHR$(254)
```

Nu får du det store Æ med 0 og det lille med 1. Læg mærke til, at de små bogstaver har to linjer luft foroven. Ofte vil det være en god idé at definere de nye karakterer, så de passer i maskinens "stil". Vi kunne også have haft et æ som

```
*****
  * *
*****
* *
*****
```

men det ville ikke rigtigt have "passet". Definer selv Å og å. Disse bogstaver samt hjælpetaster som den, der holder orden på farverne, kan du selvfølgelig have på et bånd, som du LOADer ind, hver gang du skal bruge maskinen, din egen private ROM. Når du har kørt programmet, sletter du det med NEW (så du kan gå videre med et andet program), men Æ og æ vil stadig kunne kaldes frem med tasterne 0 og 1.

Definerbare karakterer kan bruges til mangt og meget. Et problem kan det f.eks. være, at maskinen mangler sprites. En sprite har to egenskaber, som karakteren ikke har:

1. Den er større
2. Den bevæger sig fuldstændig glidende i stedet for, som karakterer, at springe et helt felt ad gangen.

For at tage det sidste problem først, kan du prøve følgende program:

```
10 FOR a = 1 TO 8 : READ a(a) : NEXT : FOR a = 1
   TO 8 : FOR b = 1 TO 8 : a$ = BIN$(a(b), 8) : a$ =
```

```
    RIGHT$(a$ , 1)+LEFT$(a$ , 7):c=0:FOR d=1
    TO 8:c=c+VAL(MID$(a$, d , 1))
    *2↑(8-d):NEXT:a(b)=c:SYMBOL a+239
    , a(1), a(2), a(3), a(4), a(5), a(6), a(7), a(8):
    NEXT b, a:MODE 0:BORDER 2:INK 0, 26:
    INK 1, 6
20  FOR a=1 TO 8:LOCATE 1, 13:FOR b=1 TO
    20:PRINT CHR$(a+239);:NEXT b, a:GOTO
    20
30  DATA 0, 24, 60, 126, 126, 60, 24, 0
```

Kør programmet, og du vil se en række røde bolde bevæge sig fuldstændig glidende over skærmen. De bevæger sig altså ikke en plads ad gangen, men en prik.

Der er ikke noget særlig mystisk ved programmet. Alt, hvad du behøver, er at udskifte de 8 tal, der danner bolden, i linje 30 med 8 andre.

Bruger du f.eks. vores Æ-tal, er det en række Æer, der sejler over skærmen. Princippet kan anvendes til at lade dine rumskibe, små mænd osv., bevæge sig lige så tegnefilmsagtigt som på de virkelig dyre spil. Læg mærke til

```
BIN$(a(b), 8)
```

Tallet 8 angiver her, at vi ønsker det binære tal med 8 cifre.

```
BIN$(27)
```

er altså stadig

```
11011
```

mens

```
BIN$(27 , 8)
```

er

00011011

Pointen er naturligvis, at hver linje i karakteren rykker en tak til højre på grund af

```
a$=RIGHT$(a$, 1)+LEFT$(a$, 7)
```

det princip, vi også anvendte i lysavisen. Resten er omregninger fra et system til et andet.

Vores andet problem er karakterens størrelse. Det kan vi imidlertid klare ved at sætte flere karakterer sammen og lade dem flytte samtidig. For eksempel kan man, hvis man – som jeg – har en svigermor, man har svært ved at undvære længere tid ad gangen, klare sig med dette program:

```
10 a$=CHR$(143):b$=CHR$(233):FOR a=1 TO
   10:FOR b=1 TO 8:READ a(b):NEXT:
   SYMBOL a+239,a(1),a(2),a(3),a(4),a(5),a(6),
   a(7),a(8):NEXT:MODE 0:BORDER 19:INK
   0,11:INK 1,16:INK 2,0:PAPER 0:PEN
   1:FOR a=1 TO 500:PRINT CHR$(228);:
   NEXT
20 PEN 2:LOCATE 8,10:PRINT b$;b$;b$;b$;
   b$:LOCATE 8,11:PRINT b$;a$;a$;a$;b$:
   LOCATE 8,12:PRINT b$;CHR$(240);CHR$
   (241);CHR$(242);b$
30 LOCATE 8,13:PRINT b$;CHR$(243);CHR$
   (244);CHR$(245);b$:LOCATE 8,14:PRINT
   b$;CHR$(246);CHR$(247);CHR$(248);b$:
   LOCATE 8,15:PRINT b$;b$;b$;b$;b$
40 LOCATE 10,14:PRINT CHR$(247):IF
   RND<0.9 THEN 40
```

```
50 LOCATE 10 , 14 : PRINT CHR$(249) : PRINT
   CHR$(7) : IF RND<0.9 THEN 50
60 GOTO 40
70 DATA 255 , 240 , 224 , 199 , 192 , 199 , 200 , 242 ,
   255 , , , , , , 129 , 126 , 255 , 15 , 7 , 227 , 3 , 227 , 19 ,
   79 , 136 , 135 , 128 , 128 , 192 , 64 , 32 , 16 , 129 , 36 ,
   36 , 36 , 102 , 66 , 126 , , 17 , 225 , 1 , 1 , 3 , 2 , 4 , 8 ,
   8 , 4 , 2 , 2 , 2 , 3 , , , , 255 , 129 , 255 , , 255 , , , 16 ,
   32 , 64 , 64 , 64 , 192 , , , , 255 , 129 , 129 , 255 , ,
   255 , ,
```

eller

10 Karakterer indlæses fra DATA. Farver. Tapet.
20 Øverste del af portræt.
30 Nederste del af portræt.
40 Første del af bevægelse.
50 Anden del af bevægelse.
60 Ny bevægelse.
70 Data.

Vi har faktisk her et "portræt" på 3 gange 4 karakterer. På samme måde kan du opbygge et fartøj eller eventuelt ved at gentage karakterer et helt landskab.

Prøv

```
10 FOR a=1 TO 8 : READ a(a) : NEXT
30 SYMBOL 255 , a(1) , a(2) , a(3) , a(4) , a(5) , a(6) , a(7) ,
   a(8) : MODE 0 : FOR a=1 TO 500 : PRINT CHR$
   (255) ; : NEXT
40 GOTO 40
50 DATA 126 , 218 , 216 , 252 , 216 , 218 , 222 , 0
```

En omstændelig måde at skrive skærmen fuld af Æer på? Jo, men den giver os mulighed for at eksperimentere lidt.

Indføj

```
20 FOR a=1 TO 8 : a(a)=255 – a(a) : NEXT
```

Karakteren bliver "negativ", for eksempel blå på gul i stedet for gul på blå. Eller prøv

```
20 FOR a=1 TO 8 : b(a)=a(9 – a) : NEXT : FOR a=1
   TO 8 : a(a)=b(a) : NEXT
```

Æer på hovedet! Hvad med spejlvendt:

```
20 FOR a=1 TO 8 : a$=BIN$(a(a), 8) : b=0 : FOR
   c=1 TO 8 : b=b+VAL(MID$(a$ , c , 1))*2↑(c – 1) :
   NEXT : a(a)=b : NEXT
```

Vi kan også få lidt større bogstaver:

```
10 MODE 2
20 FOR a=1 TO 8 : READ a(a) : FOR b=1 TO 9 :
   FOR c=1 TO 8 : IF VAL(MID$(BIN$(a(a), 8), c , 1))
   THEN PRINT CHR$(143) ; : GOTO 40
30 PRINT "m" ;
40 NEXT c , b : PRINT : NEXT : RESTORE : GOTO 20
50 DATA 126 , 218 , 216 , 252 , 216 , 218 , 222 , 0
```

Alle eksemplerne fungerer selvfølgelig på alle karakterer. Det er bare et spørgsmål om at skifte DATA-tallene ud. Disse eksempler har hovedsagelig til formål at sætte dig i gang med at eksperimentere selv.

Eksperimenterne er det vigtigste. At sætte sig det mål at få noget ganske bestemt frem på skærmen, og så finde ud af, hvordan det lader sig gøre.

For mange mennesker er leg med datamater tidsspilde.

De har en meget primitiv opfattelse af, hvad viden er, de tror, det er noget med at lære udenad.

Eller måske foragter de eller er bange for viden. Mange synes i dag i fuldt alvor at mene, at man ikke både kan være intelligent og et sympatisk menneske, måske fordi vores samfund er ved at udvikle sig derhen, hvor man skal være snotdum for ikke at protestere.

Men der er nogle af os, der stadig har vores nysgerrighed i behold, som ikke er tilfredse med remser og paroler, men vil opdage verden selv. Vi er ikke tilfredse med patentløsningerne, vi vil finde vores egne løsninger, og derfor kaster vi os – mere eller mindre bevidst – over alt, hvad der kan stimulere og optræne evnen til problemløsning. Her er datamaten uovertruffen.

Det er ikke mit formål med snart en hel række bøger at udklække elektroniske flipproletarer, men mennesker med deres egne løsninger i stedet for samfundets, politikerne og deres stråmænd, de politiske "aktivister". Derfor er det måske også i en forstand ligegyldigt, om jeg lærer dem BASIC eller hottentot.

Nu er BASIC jo imidlertid kolossalt udbredt. Enhver kiosk har stakkevis af blade fyldt med programlister, der ofte er den første indgang.

Dette kræver imidlertid også, at de enkelte fabrikater ikke i deres konkurrence barrikaderer sig for meget imod hinanden. Idealet må være, at alle brugere, uanset deres maskines varemærke, kan udveksle programmer og erfaringer.

Det har altid været en kilde til ærgrelse for mig, at skærmbehandlingen – altså, når vi skal i gang med de flyvende tallerkener – varierer så meget fra maskine til maskine. Sådan har det imidlertid altid været.

Noget andet er så, at nogle mærker er begyndt at køre med en udvidet BASIC. Dette kan være en stor fordel.

Hvem ville undvære RENUMBER, som f.eks. ikke findes

på COMMODOREs maskiner? Mindre nyttigt er et påfund som ELSE. Vist er da

```
10 INPUT a
20 IF a=1 THEN PRINT "OK" ELSE PRINT
   "NEJ": RUN
```

elegantere end

```
10 INPUT a
20 IF a=1 THEN PRINT "OK": END
30 PRINT "NEJ": RUN
```

eller sågar

```
10 INPUT a
20 IF a<>1 THEN PRINT "NEJ": GOTO 10
30 PRINT "OK"
```

men hvis prisen så er, at programmet ikke kan køre på f.eks. en COMMODORE 64, er det – efter min mening – for dyrt betalt. Det er selvfølgelig i høj grad en personlig vurdering. Ingen ville fornuftigvis udelade DATA af sine programmer, fordi ZX81 ikke har dette reserverede ord i sit vokabularium.

Alt dette betyder naturligvis ikke, at jeg ikke vil gennemgå WHILE, WEND, AFTER, og hvad det altsammen hedder. Faktisk er det det, resten af kapitlet handler om.

Det betyder imidlertid, at jeg har ventet længst muligt med at introducere disse praktiske, men overflødige funktioner. Dermed har jeg givet læseren frit valg.

Efter min gennemgang af dem kan han sporenstregs begynde at inkorporere dem i sine programmer. Men han har også lært at programmere uden dem, hvis han sætter de intermaskinelle relationer højere. Et eksempel er

INSTR

Kan du huske

```
10 a$="KØBENHAVN":FOR a=1 TO 8:IF
   MID$(a$,a,2)="EN" THEN PRINT "ok"
20 NEXT
```

Det kan vi lave om til

```
10 PRINT INSTR("KØBENHAVN","EN")
```

og så får vi oven i købet tallet 4: EN begynder i KØBENHAVNs 4. karakter. Prøv

```
10 PRINT INSTR("KØBENHAVN","OS")
```

Vi får 0, eftersom OS ikke forekommer. Prøv så

```
10 PRINT INSTR(5,"KØBENHAVN","EN")
```

Maskinen fandt ikke EN, for 5-tallet betød, at den skulle starte sin eftersøgning med 5. karakter. Vores strudseprogram kan blive til

```
10 INPUT "STØRSTE FUGL";a$:IF INSTR(a$,
   "struds") THEN END
20 RUN
```

eller

```
10 INPUT "STØRSTE FUGL";a$:IF INSTR(a$,
   "struds") THEN END ELSE RUN
```

De følgende eksempler i kapitlet om strengdeling kan blive til

```
10 INPUT "Delstreng"; a$: PRINT INSTR
   ("KØBENHAVN", a$)
```

og

```
10 INPUT "Streng"; a$: INPUT "Delstreng"; b$:
   PRINT INSTR(a$, b$)
```

WHILE-WEND er en løkke ligesom FOR-NEXT, men i stedet for at tælle omgange, undersøger den, om en betingelse er opfyldt.

```
10 WHILE a$ <> "jupiter"
20 INPUT "STØRSTE PLANET"; a$
30 WEND
```

kan vi oversætte: Mens (så længe) a$ ikke er "jupiter", skal maskinen blive ved med at modtage nye forsøg. WEND "lukker" løkken, ligesom NEXT i FOR-NEXT-løkken.

AFTER

angiver, at et vist tidsrum skal hengå, før en subrutine går i gang. Prøv

```
10 AFTER 500 GOSUB 40
20 PRINT "PROGRAM"
30 GOTO 30
40 PRINT "PROGRAMMET HAR KØRT I 10
   SEKUNDER"
50 RETURN
```

Læg mærke til, at programmet lynhurtigt går ind i den uendelige løkke 30 GOTO 30, men det gør ingenting: Vækkeuret er stillet! Faktisk har maskinen fire sådanne vækkeure, vækkeur 0, 1, 2 og 3. I eksemplet

```
10 AFTER 500 GOSUB 50
20 AFTER 3000,1 GOSUB 70
30 PRINT "PROGRAM"
40 GOTO 40
50 PRINT "PROGRAMMET HAR KØRT I 10
   SEKUNDER"
60 RETURN
70 PRINT "OG NU I ET HELT MINUT"
80 RETURN
```

har vi sat vækkeur 0 (nul er underforstået) til at vække efter 10 sekunder, og vækkeur 1 til at vække efter 60. For det tilfældes skyld, at vi skulle blive utålmodige, kan vi udskifte 40 med

```
40 a$=INKEY$: IF a$<>"" THEN PRINT REMAIN
   (0), REMAIN(1): END
41 GOTO 40
```

```
REMAIN(0)
```

er, som du vil se, det, der er tilbage på vækkeur 0. Prøv

```
? REMAIN(2)
```

Vi får 0, for vækkeur 2 er slet ikke stillet. Kør det reviderede program, og bed derefter om

```
? REMAIN(0), REMAIN(1)
```

Det er også 0 nu, for REMAIN standser uret.

En variant er

EVERY

Prøv

```
10 EVERY 50 GOSUB 30
20 GOTO 20
30 PRINT CHR$(7);
40 RETURN
```

REM

findes på næsten alle maskiner. Dette underlige ord betyder, at maskinen *ikke* skal gøre det, der kommer efter i samme linje. Det gør, at vi kan have forklarende linjer i selve programmet. I vores UFO-spil kunne vi f.eks. have

```
 15 REM HOVEDLØKKE
 35 REM BASE
 55 REM RAKETRUTINE
 75 REM BOMBERUTINE
115 REM SLUT
```

AUTO

betyder, at maskinen selv sætter linjenumrene.

AUTO 100 , 5

betyder, at den begynder med linjenummer 100 og tæller 5 frem for hver ny linje. Har du stadig EVERY-programmet i maskinen? Skriv

```
AUTO
```

Hermed er underforstået, at første linje skal have linjenummer 10, og at der skal være 10 imellem hver. Men hvad sker der, når du begynder at skrive?

Maskinen skriver små stjerner ud for de fire første linjer. På den måde advarer den om, at der allerede eksisterer en linje 10, 20, etc.

Vi kan også lave et tekstvindue i f.eks. en grafisk afbildning. Prøv

```
10 MODE 2 : INK 0 , 26 : INK 1 , 0 : EVERY 100
   GOSUB 50 : WINDOW 20 , 60 , 10 , 20
20 PRINT "Dette er et tekstvindue. Tal nr 1 (20) i
   WINDOW-ordren angiver nu, hvor langt til venstre
   vinduet starter, nr 2 (60), hvor langt hen det kommer,
   endelig siger nr 3 (10), hvor langt op, og det sidste
   (20),"
30 PRINT "hvor langt ned. Farver kan give et endnu
   klarere vindue. Teksten skal naturligvis redigeres for
   at passe i vinduet. Derfor er gevinsten ved brug af
   WINDOW nok ikke videre stor, men sjovt er det da.
   Ikke?!"
40 GOTO 40
50 INK 1 , INT (RND * 13) : RETURN
```

Med

```
PEEK
```

kan vi få indholdet af en hvilken som helst adresse. Vi ved, at en byte kan oplagre et tal mellem 0 og 255, og at alt, hvad maskinen skal huske, er oplagret på den måde. Prøv

```
10 PRINT "AMSTRAD"
20 PRINT : PRINT PEEK(370); : FOR a=371 TO 383:
   PRINT CHR$(PEEK(a)); : NEXT
30 PRINT : PRINT : LIST 10 : PRINT
```

Kør det. Vi får

```
AMSTRAD

10 Ω "AMSTRAD"

10 PRINT "AMSTRAD"
```

Linje 10 skriver AMSTRAD, og linje 30 LISTer linje 10. Men hvad gør linje 20?

Den LISTer faktisk også linje 10, men på sin helt egen måde. Den er nemlig brudt ind i programfilen, der opbevarer programmet (det er altså den, der nulstilles, når vi siger NEW) og har læst det direkte. Omega betyder altså (for maskinen) PRINT. Skift ud med

```
10 REM "AMSTRAD"
```

Vi får > – eller rettere sagt 62. Prøv nu

```
10 PRINT "AMSTRAD"
20 POKE 375,77 : POKE 376,48 : POKE 377,82 :
   POKE 378,66
30 LIST
```

Kør det. Maskinen siger

```
AMSTRAD
10 PRINT "MØRBRAD"
```

```
20 POKE 375 , 77 : POKE 376 , 48 : POKE 377 , 82 :
   POKE 378 , 66
30 LIST
```

POKE er den "aktive" PEEK.

```
POKE a , b
```

betyder: Gå ind i adresse a og skift indholdet ud med b. I det foreliggende eksempel har vi nok så frækt rettet direkte i programfilen. Kør programmet igen. Den *skriver* virkelig MØRBRAD. En lidt mere omstændelig måde at rette i sine programmer på.

Prøv

```
? PRINT MAX(1 , 2 , 3 , 4)
```

Maskinen skriver 4, for vi har bedt den skrive det største af tallene, 1, 2, 3 og 4. Tilsvarende vil

```
? PRINT MIN(1 , 2 , 3 , 4)
```

give 1. Hvad skulle vi dog have gjort uden den funktion! Andre er ganske praktiske. I stedet for 17 mellemrum i en PRINT-linje kan vi skrive f.eks.

```
PRINT "AMSTRAD" ; SPACE$(17) ; "AMSTRAD"
```

Skriv

```
10 FOR a = 1 TO 10 : PRINT a , : NEXT
```

Maskinen skriver (i MODE 1)

```
1              2              3
```

```
4            5            6
7            8            9
10
```

dvs. maskinen flytter 13 for hvert komma, hver kolonne (zone) er, uanset hvor meget der står i den, 13 karakterer bred. Prøv nu

```
10 ZONE 7 : FOR a = 1 TO 10 : PRINT a , : NEXT
```

Maskinen har naturligvis også en del matematiske funktioner. Endelig kan du lave dem selv med

```
DEF FN
```

Prøv

```
10 DEF FNa(b) = 1 / b
20 PRINT FNa(9)
```

Vi har her defineret en helt ny funktion, der tager reciprokværdien af et tal (1 divideret med tallet). Funktionen hedder a, mens b er en "pseudovariabel", der fortæller maskinen, hvad der skal ske med funktionens argument. Vi kunne også kalde den REC (for RECiprok):

```
10 DEF FNREC(a) = 1 / a
20 PRINT FNREC(9)
```

Vi behøver strengt taget intet argument. Prøv

```
10 DEF FNa = INT(RND * 6) + 1 : DEF FNb = INT
   (RND * 6) + INT(RND * 6) + 2
20 PRINT FNa , FNb : GOTO 20
```

FNa slår med en terning, FNb med to.

9. LYD

SOUND – ENV – ENT – JOY

Skru helt op for volumen. Prøv

```
10 SOUND 1,119
```

Dette er så nogenlunde den simpleste musik-kommando, du kan give AMSTRAD. Den betyder simpelt hen, at stemme eller kanal nummer 1 skal spille tonen med toneværdi 119.

Der er i alt tre stemmer på AMSTRAD, hvilket gør det muligt at spille op til tre toner på en gang. De hedder 1, 2 og 4 (bit 0 – regnet bagfra – bit 1 og bit 2 tændt). Her er nogle vigtige toneværdier:

C	239	119	60	30
CIS	225	113	56	28
D	213	106	53	27
DIS	201	100	50	25
E	190	95	47	24
F	179	89	45	22
FIS	169	84	42	21
G	159	80	40	20
GIS	150	75	38	19
A	142	71	36	18
AIS	134	67	34	17
H	127	63	32	16

Jo lavere tallet, jo højere tonen. De fire tal ud for C er således C i fire oktaver. Prøv

```
10 SOUND 1 , 239
20 SOUND 1 , 119
30 SOUND 1 , 60
40 SOUND 1 , 30
```

og

```
10 FOR a=1 TO 13 : READ b : SOUND 1 , b : NEXT
20 DATA 119 , 113 , 106 , 100 , 95 , 89 , 84 , 80 , 75 , 71 ,
   67 , 63 , 60
```

```
10 FOR a=1 TO 8 : READ a(a) : NEXT
20 a$=INKEY$ : IF a$="" THEN 20
30 a=ASC(a$)-48 : IF a<1 OR a>8 THEN 20
40 SOUND 1 , a(a) : GOTO 20
50 DATA 119 , 106 , 95 , 89 , 80 , 71 , 63 , 60
```

Vi har her et lille klaver, som vi kan spille "Dejlig er den himmel blå" på. Tasterne 1-8 er vores klaviatur. Tilføj selv flere toner.

Måske synes vi, tonerne er lidt for korte. Vi kan så tilføje et tredje tal, der angiver, hvor længe tonen skal vare i hundrededele af et sekund. Er intet nærmere specificeret, varer den 1/5 sekund svarende til tallet 20.

```
SOUND 1 , a(a)
```

og

```
SOUND 1 , a(a) , 20
```

giver altså samme resultat. Prøv imidlertid at ændre linje 40 i vores klaverprogram til

```
40 SOUND 1 , a(a) , 30 : GOTO 20
```

Læg mærke til, at du nu selvfølgelig ikke kan spille så hurtigt på "klaveret". Hver enkelt tone skal have tid til at klinge ud.

Noget rigtigt klaver bliver AMSTRAD selvfølgelig aldrig til (hvis vi ville have haft et klaver, havde vi købt et klaver). Derimod egner maskinen sig fortrinligt til at spille små melodistumper i f.eks. spil, som du i forvejen har kodet ind i maskinen via DATA-linjer. I dette kapitel skal vi se på, hvordan det lader sig gøre.

Hvad enten du kender Chopins "Marche Funèbre" eller ej, er jeg sikker på, at du vil genkende den. Det er den, spilleautomaten spiller, når du er død og ikke får mere for den tikrone. Det lyder omtrent sådan her:

```
10 FOR a=1 TO 11:READ b,c:SOUND 1,b,c*
   10:FOR d=1 TO c*100:NEXT d,a
20 DATA 119,8,119,8,119,1,119,8,100,4,
   106,4,106,4,119,4,119,8,127,1,119,8
```

Læg mærke til, at melodien kun har fire toner. Hvis vi imidlertid bare spillede dem efter hinanden uden at tage højde for deres respektive varighed, ville der ikke være meget melodi.

Vi må altså indlæse to tal for hver tone, nemlig dens *toneværdi* (b) og dens *varighed* (c). Bemærk også, at tonernes indbyrdes varighed ikke har noget at gøre med det tempo, vi spiller melodien i. Tempoet afgøres af det tal, vi ganger c med (i dette tilfælde 10).

Vi kan ændre dette tal og få et andet tempo, men så må vi også ændre pausen, der dannes af d-løkken og giver hver tone tid til at klinge ud. Vi ved (fra fjerde kapitel), at en FOR-NEXT-løkke på 1000 omdrejninger svarer omtrent til 1 sekund.

Når vi så yderligere ved, at tonens varighed måles i hundrededele af et sekund, er det ikke vanskeligt at indse,

at denne FOR-NEXT-løkke altid skal have et antal omdrejninger cirka svarende til ti gange tonens varighed. Et lidt morsommere eksempel er måske

```
10 FOR a = 1 TO 33 : READ b , c : SOUND 1 , b , c *
   15 , 15 : FOR d = 1 TO c * 150 : NEXT d , a
20 DATA 119 , 4 , 119 , 1 , 179 , 1 , 119 , 1 , 119 , 8 , 89 ,
   4 , 119 , 4 , 179 , 8 , 119 , 4 , 119 , 1 , 179 , 1 , 119 , 1 ,
   119 , 8 , 142 , 3 , 159 , 1 , 159 , 12 , 119 , 4 , 119 , 1 ,
   179 , 1 , 119 , 1 , 119 , 8 , 89 , 4 , 119 , 4 , 179 , 8 ,
   179 , 1 , 179 , 1 , 179 , 4 , 159 , 1 , 159 , 1 , 159 , 4 ,
   106 , 2 , 119 , 1 , 89 , 2
```

Vi har her fået et *fjerde* tal på vores SOUND-kommando: 15, der hentyder til lydstyrken, der går fra 0 til 15 (den er normalt 12). Prøv at rette til

```
SOUND 1 , b * 2 , c * 15 , 15
```

og

```
SOUND 1 , CINT(b / 2) , c * 15 , 15
```

Prøv også

```
10 FOR a = 1 TO 63 : READ b , c : SOUND 1 , b , c *
   15 , 15 : FOR d = 1 TO c * 150 : NEXT d , a
20 DATA 142 , 2 , 142 , 1 , 142 , 1 , 142 , 2 , 142 , 1 ,
   142 , 1 , 142 , 2 , 213 , 2 , 179 , 2 , 142 , 2
30 DATA 159 , 2 , 159 , 1 , 159 , 1 , 159 , 2 , 159 , 1 ,
   159 , 1 , 159 , 2 , 239 , 2 , 190 , 2 , 159 , 2 , 142 , 2 ,
   142 , 1 , 142 , 1
40 DATA 142 , 2 , 142 , 1 , 142 , 1 , 142 , 2 , 127 , 2 ,
   119 , 2 , 106 , 2 , 119 , 2 , 142 , 2 , 159 , 2 , 190 , 2 ,
   213 , 4 , 213 , 4
```

```
50 DATA 142 , 4 , 142 , 3 , 142 , 1 , 142 , 2 , 213 , 2 ,
   179 , 2 , 142 , 2 , 159 , 4 , 159 , 3 , 159 , 1 , 159 , 2 ,
   239 , 2 , 190 , 2 , 159 , 2
60 DATA 142 , 4 , 142 , 3 , 142 , 1 , 142 , 2 , 127 , 2 ,
   119 , 2 , 106 , 2 , 119 , 2 , 142 , 2 , 159 , 2 , 190 , 2 ,
   213 , 4 , 213 , 4
```

Denne gamle shanty har faktisk en andenstemme, som vi passende kan indføje:

```
10 FOR a=1 TO 63 : READ b , c , d : SOUND 1 , b ,
   d*15 , 15 : SOUND 2 , c , d*15 , 15 : FOR e=1 TO
   d*150 : NEXT e , a
20 DATA 142 , 179 , 2 , 142 , 179 , 1 , 142 , 179 , 1 ,
   142 , 179 , 2 , 142 , 179 , 1 , 142 , 179 , 1 , 142 , 179 ,
   2 , 213 , , 2 , 179 , , 2 , 142 , , 2
30 DATA 159 , 190 , 2 , 159 , 190 , 1 , 159 , 190 , 1 ,
   159 , 190 , 2 , 159 , 190 , 1 , 159 , 190 , 1 , 159 , 190 ,
   2 , 239 , , 2 , 190 , , 2 , 159 , , 2 , 142 , 179 , 2 , 142 , 179 ,
   1 , 142 , 179 , 1
40 DATA 142 , 179 , 2 , 142 , 179 , 1 , 142 , 179 , 1 ,
   142 , 179 , 2 , 127 , 190 , 2 , 119 , 142 , 2 , 106 , 127 ,
   2 , 119 , 142 , 2 , 142 , 179 , 2 , 159 , 190 , 2 , 190 ,
   239 , 2 , 213 , , 4 , 213 , , 4
50 DATA 142 , 179 , 4 , 142 , 179 , 3 , 142 , 179 , 1 ,
   142 , 179 , 2 , 213 , , 2 , 179 , , 2 , 142 , , 2 , 159 , 190 ,
   4 , 159 , 190 , 3 , 159 , 190 , 1 , 159 , 190 , 2 , 239 , ,
   2 , 190 , , 2 , 159 , , 2
60 DATA 142 , 179 , 4 , 142 , 179 , 3 , 142 , 179 , 1 ,
   142 , 179 , 2 , 127 , 159 , 2 , 119 , 142 , 2 , 106 , 127 ,
   2 , 119 , 142 , 2 , 142 , 179 , 2 , 159 , 190 , 2 , 190 ,
   239 , 2 , 213 , , 4 , 213 , , 4
```

En andenstemme har selvfølgelig aldrig en tone for hver tone i førstestemmen, der bliver pauser i andenstemmen,

hvor førstestemmen spiller alene. Dette problem løser AMSTRAD BASIC med en række uhyre komplicerede funktioner.

Som man vil se af ovenstående eksempel er de også overflødige. Vi efterlader simpelt hen et hul (egentlig et nul) der, hvor pauserne skal være. En pause i musikken klarer vi med nul i begge stemmer og en varighed af den rigtige størrelse.

Noget andet er, at du måske ikke synes, det lyder særlig godt. Tonerne er ligesom for "rå". Kan vi gøre noget ved det? Prøv

```
10 SOUND 1 , 239 , 150 , 15
```

Intet mystisk ved det: Et C i kanal 1, der spiller halvandet sekund med lydstyrke 15. Men prøv så

```
 5 ENV 1 , 15 , - 1 , 10
10 SOUND 1 , 239 , 150 , 15 , 1
```

Ja, for det første er der føjet et *femte* tal til SOUND, et 1-tal. Dette ettal betyder: Brug ENV nummer 1! Nå. Hvor køber man sådan en?

ENV står for ENvelope Volume, det er lydstyrkens "form". Hermed menes, at lydstyrken ikke behøver at være den samme toneforløbet igennem. Ovenover står så ENV 1 defineret.

```
ENV 1 , 15 , - 1 , 10
```

betyder så noget som

ENV *1* skal være defineret sådan, at toneforløbet inddeles i *15* trin. For hvert trin skal lydstyrken vokse med –1 (hvilket

i realiteten vil sige, at den falder med 1). Endelig skal hvert trin vare *10* hundrededele af et sekund eller 1/10 sekund.

Læg mærke til, at antallet af trin ganget med hvert enkelt trins varighed gerne skal svare til hele tonens varighed. I vores tilfælde skal 15 gange 10 altså blive 150. Den lydstyrke, som er angivet i SOUND-kommandoen bliver da et *udgangspunkt* for ENV, og vi får forløbet

1. tiendedel sek: lydstyrke 14
2. tiendedel sek: lydstyrke 13
3. tiendedel sek: lydstyrke 12
. . . .
15. tiendedel sek: lydstyrke 0

Lidt mindre matematisk udtrykt får vi en tone, der slås an og klinger ud, omtrent som en klokke. Prøv selv at høre efter. Prøv så

```
 5 ENV 1 , 15 , 1 , 10
10 SOUND 1 , 239 , 150 , 0 , 1
20 GOTO 10
```

Her stiger lydstyrken tværtimod, og vi får noget, der minder om tonen fra en violin. Vi kan også have

```
 5 ENV 1 , 15 , 1 , 5 , 15 , – 1 , 5
10 SOUND 1 , 239 , 150 , 0 , 1
20 GOTO 10
```

Denne ENV-ordre er i virkeligheden to i en. Den siger:

Definer ENV sådan, at den først stiger i trin af 5/100 sekunders varighed (15 , 1 , 5) og derefter falder med samme hastighed (15 , – 1 , 5).

Vi kan sige, at ENV-kommandoen her falder i to *sektioner*. Faktisk kan vi have op til fem sådanne afsnit, i hvilket sidste tilfælde antallet af enkelte tal kommer op på 1 + 5 * 3 = 16 ! Prøv

```
5 ENV 1 , 15 , 1 , 3 , 1 , 0 , 60 , 15 , – 1 , 3
```

Vi har tre sektioner svarende til ti tal. Den midterste sektion (1 , 0 , 60) består kun af 1 trin, og lydstyrken hverken stiger eller falder (0) i 60/100 sekund. Tonen stiger altså, holder sig en tid konstant, og falder så.

Læg mærke til, at vi stadig får 15 * 3 + 1 * 60 + 15 * 3 = 150. Pengene passer.

Pointen er selvfølgelig, at vi kan opnå mange forskellige virkninger på denne måde. Anvendt på the drunken sailor kunne det f.eks. blive til

```
 5 ENV 1 , 15 , – 1 , 1
10 FOR a = 1 TO 63 : READ b , c , d : SOUND 1 , b ,
   d * 15 , 15 , 1 : SOUND 2 , c , d * 15 , 11 : FOR e = 1
   TO d * 150 : NEXT e , a : RUN
```

Nu har vi fået to sømænd for de samme penge: En spiller mundharmonika, mens den anden er ganske ferm med en banjo!

Kan vi komme så langt ved at pille ved lydstyrken, kan vi naturligvis komme endnu længere ved at lave det samme nummer med tonehøjden. Lad os vende tilbage til vores C:

```
10 SOUND 1 , 239 , 150 , 15
```

Prøv nu

```
 5 ENT 1 , 15 , – 10 , 10
10 SOUND 1 , 239 , 150 , 15 , 0 , 1
```

Nu er femte plads, som er reserveret ENV tom, til gengæld har vi på *sjette* plads ENT, der læses på præcis samme måde som ENV, i det foreliggende tilfælde:

ENT *1* skal være defineret sådan, at toneforløbet inddeles i *15* trin. For hvert trin skal toneværdien vokse med *–10* (hvilket i realiteten vil sige, at den falder med 10, og tonen stiger). Endelig skal hvert trin vare *10* hundrededele af et sekund eller 1/10 sekund.

Også her bliver 15 * 10 = 150, og igen er udgangspunktet for ENT SOUND-kommandoens toneværdi. Forløbet bliver da

1. tiendedel sek: toneværdi 229
2. tiendedel sek: toneværdi 219
3. tiendedel sek: toneværdi 209

. . . .

6. tiendedel sek: toneværdi 179 (tonen F)
7. tiendedel sek: toneværdi 169 (tonen FIS)
8. tiendedel sek: toneværdi 159 (tonen G)

. . . .

12. tiendedel sek: toneværdi 119 (C en oktav længere nede)

. . . .

Den sidste tone, vi hører, er altså et F i en helt anden oktav. Det er naturligvis et underligt instrument, der opfører sig på den måde, og som regel holder man sig til mindre ud-

sving. Som lydeffekt til spil egner en ENT som denne sig naturligvis udmærket. Prøv på samme måde

```
 5 ENT 1 , 15 , 10 , 10
10 SOUND 1 , 239 , 150 , 15 , 0 , 1
20 GOTO 10
```

og

```
 5 ENT 1 , 15 , 1 , 5 , 15 , – 1 , 5
10 SOUND 1 , 239 , 150 , 15 , 0 , 1
20 GOTO 10
```

og

```
5 ENT 1 , 15 , 1 , 3 , 1 , 0 , 60 , 15 , – 1 , 3
```

Mens vi taler om lydeffekter, er der et *syvende* og sidste tal, der kan være os til nytte. Dette tal betyder simpelt hen blot støj, men kombineret med de andre kan det blive til helt rimelige effekter.

```
10 SOUND 1 , 239 , 150 , 15 , 0 , 0 , 15
```

er bare en tone med støj på, men prøv så

```
10 ENV 1 , 15 , – 1 , 5
20 ENT 1 , 15 , 1 , 5
30 SOUND 1 , 239 , 75 , 15 , 1 , 1 , 15
40 IF RND<0.999 THEN 40
50 GOTO 30
```

og vi er (næsten) i Dodge City.

```
10 ENV 1 , 15 , – 1 , 1
20 ENT 1 , 15 , 1 , 1
30 SOUND 1 , 500 , 50 , 15 , 1 , 1 , 15
40 IF RND<0.995 THEN 40
50 GOTO 30
```

Bedemanden sømmer endnu en kiste til. Gary Cooper beslutter alligevel at forlade byen inden klokken tolv:

```
10 ENV 1 , 10 , – 1 , 1 : ENT 1 , 10 , 1 , 1 : FOR a = 15
   TO 10 STEP – 1 : FOR b = 1 TO 10 : SOUND 1 ,
   1000 , 10 , a , 1 , 1 , 15 : SOUND 1 , 1000 , 10 , a , 1 ,
   1 , 15 : SOUND 1 , 1000 , INT(RND * 5) + 15 , a , 1 , 1,
   15 : NEXT b , a
```

Læg mærke til, at SOUND-kommandoens volumen-tal egentlig kun går fra 0 til 15, når vi har at gøre med ENV. I andre tilfælde går den fra 0 til 7 og begynder så forfra. Kan du huske vores UFO-spil? En flyvende tallerken bliver en hel del mere faretruende, når den lyder sådan her:

```
10 a = 100 : b = 1 : ENT 1 , 10 , 1 , 1 , 10 , – 1 , 1
20 a = a + b : IF a = 50 OR a = 150 OR RND<0 . 1
   THEN b = – b
30 SOUND 1 , a , 20 , 15 , 0 , 1 : GOTO 20
```

I spillet bruger du selvfølgelig rumskibets eget linjeflag, så lyden følger dets bevægelser. Når det skyder, kan det lyde sådan her:

```
10 ENT 1 , 5 , 1 , 1 , 5 , – 1 , 1
20 SOUND 1 , 10 , 10 , 15 , 0 , 1 : GOTO 20
```

Når De hører denne lyd, søg øjeblikkelig beskyttelse! Luftalarm:

```
10 ENT 1 , 15 , – 10 , 1 , 15 , 10 , 1
20 SOUND 1 , 250 , 30 , 15 , 0 , 1 : GOTO 20
```

Ved at eksperimentere med disse få principper kan du give dine programmer en rig og varieret lydside. Og musik og lydeffekter er jo ofte det, der giver et spil det ekstra, der gør det spændende.

En anden praktisk ting er joystick. Hvis du har sådan en, kan du tilslutte den og prøve

```
10 MODE 0
20 LOCATE 1 , 1 : PRINT JOY(0) : GOTO 20
```

Du kan nu let få værdierne

```
 1 OP (BIT 0 TÆNDT)
 2 NED (BIT 1 TÆNDT)
 4 VENSTRE (BIT 2 TÆNDT)
 8 HØJRE (BIT 3 TÆNDT)
16 FYR (BIT 4 TÆNDT)
```

Hvis du nu fører joysticken op til venstre, mens du trykker på affyringsknappen, vil du få 21 (1 + 4 + 16). Disse værdier er forholdsvis lette at inkorporere i et program, f.eks. vores tegneprogram fra kapitel 6:

```
10 MODE 0 : PLOT 320 , 200
20 a = JOY(0) : DRAWR a = 4 , a = 2 : DRAWR – (a = 8)
   , – (a = 1) : MOVER XPOS = 640 , YPOS = 400 :
   MOVER – (XPOS = 0) , – (YPOS = 0) : GOTO 20
```

Her har vi jo imidlertid ikke udnyttet joystickens mulighed for at flytte på skrå. Prøv

```
10 MODE 0
20 a=JOY(0):a$=BIN$(a,4):LOCATE 1,1:PRINT
   a$: GOTO 20
```

Vi kan nu lave vores program om til

```
10 a=320:b=200:MODE 0:PLOT a,b
20 c=JOY(0):a$=BIN$(c,4):FOR c=1 TO 4:a(c)=
   VAL(MID$(a$,c,1)):NEXT:a=a+a(1)-a(2):b=
   b-a(3)+a(4):a=a+(a=640)-(a=0):b=b+(b=
   400)-(b=0):DRAW a,b:GOTO 20
```

Mange mennesker er i dag bange for datamater. Eller måske er det den ny tid, de er bange for, som datamaterne er et udtryk for.

Det er en tid, hvor dogmer falder, en oplysningstid. Per Degn ser sin autoritet blegne, Jorden var alligevel ikke flad.

Fysikere er ved at finde ud af, at verden er skabt af menneskets tænkning, at alt "fysisk" i virkeligheden er psykologisk. Biologerne prøver nu for alvor at finde ud af, hvad "psykologisk" er.

Som i enhver oplysningsperiode sættes mennesket i centrum, og det gør det aldeles skrækslagent. Det kan ikke længere gemme sig bag overtro og politik, moral og pæne manerer, det må tænke selv.

Verden viser sig pludselig at være utrolig kompliceret, vi må træne vores intellekt til det yderste og sluge så meget viden som muligt, før vi kan tillade os at have en mening om den. Ved at programmere øver vi os i at tænke, opstille, formulere og løse problemer.

Selvfølgelig vil der altid være dem, der nok mener at vide, at Jorden er flad, det siger præsten, og han er et anstændigt menneske. Datamater og alskens lærdom har fanden skabt for at forføre menneskene. Her må vi beklageligvis nok lade stå, hvo ej kan løbe.

For at kunne læse og skrive må vi først lære alfabetet. Det er denne bogs formål.

Generelle indføringer i BASIC tror jeg ikke på. Det må hedde: Du skal trykke på den knap der, og det kræver en bog lige netop til din maskine, som altså er den udmærkede AMSTRAD.

Vi har stiftet bekendtskab med variabler og programmer, funktioner og logiske operationer. Vi har indøvet grundlæggende figurer som IF-THEN og FOR-NEXT.

Vi har arbejdet med strengdeling og dimensionering og grafisk afbildning. Der er masser af spil i bogen, eller snarere en grundig gennemgang af alle de principper, du får brug for, når du skal lave nogle.

At det er tidsspilde i længden at indtaste og spille spil fremstillet af andre, kan jeg naturligvis ikke være uenig i, men selv at fremstille spil er noget af det morsomste og mest lærerige, man kan bruge sin datamat til. Jeg vil vove at påstå, at den, der grundigt studerer bogen, kan lave alle de invasioner af rumuhyrer, han orker, samt alle de andre variationer af hasardspil, mastermind, simulationer, adventures, og så videre, komplette med definerede karakterer, musik, lydeffekter, joystick, etc. etc.

"Nyttige" programmer forekommer også, men det egentlig nyttige er naturligvis at lære at programmere, hvilket har en slående lighed med at lære at tænke.

Og det er jo ikke så dårligt at kunne.

10. LABYRINTEN OG HOTELDETEKTIVEN

```
10  REM***Labyrint***
20  REM***Niveau***
30  MODE 0 : BORDER 1 : INK 0 , 1 : INK 1 , 24 :
    PAPER 0 : PEN 1 : PRINT "LABYRINT" : PRINT :
    PRINT "NIVEAU (1 – 9)?"
40  a$= INKEY$ : IF a$= "" THEN 40
50  a = ASC(a$) : IF a<49 OR a>57 THEN 40
60  REM***Startstilling***
70  m =(a – 48) / 10 : DIM a$(109) : DIM f(99) : DIM
    g(33) : DIM h(33) : FOR a = 1 TO 33 : READ g(a),
    h(a) : NEXT : FOR a = 1 TO 9 : READ i(a) : NEXT :
    FOR a = 0 TO 10 : READ j(a), k(a) : NEXT : ENV 1 ,
    10 , – 1 , 1 : ENT 1 , 10 , 1 , 1
80  SYMBOL 254 , 255 , 195 , 129 , 195 , 231 , 231 , 195 ,
    255 : SYMBOL 255 , 0 , 64 , 224 , 191 , 234 , 64 :
    BORDER 13 : FOR a = 0 TO 12 : INK a , a : NEXT :
    INK 13 , 26 : PAPER 13 : FOR a = 0 TO 109 : a$(a)
    = "mmmmm" : NEXT : FOR a = 10 TO 99 : a$(a)=
    "1111" : IF a<20 THEN MID$(a$(a), 1 , 1)= "0"
90  IF a>89 THEN MID$(a$(a), 3 , 1)= "0"
100 IF a / 10 = INT(a / 10) THEN MID$(a$(a), 4 , 1)=
    "0"
110 IF (a + 1) / 10 = INT((a + 1) / 10) THEN MID$(a$
    (a), 2 , 1)= "0"
120 f(a)= INT(RND*13) : NEXT : FOR a = 10 TO 99 :
    IF RND<0.1 THEN MID$(a$(a), 1 , 1)= "0" :
    MID$(a$(a – 10), 3 , 1)= "0"
```

```
130 IF RND<0.1 THEN MID$(a$(a), 2, 1)="0":
    MID$(a$(a+1), 4, 1)="0"
140 IF RND<0.1 THEN MID$(a$(a), 3, 1)="0":
    MID$(a$(a+10), 1, 1)="0"
150 IF RND<0.1 THEN MID$(a$(a), 4, 1)="0":
    MID$(a$(a-1), 2, 1)="0"
160 NEXT: a$(10)="1110": MID$(a$(11), 4, 1)="1":
    MID$(a$(20), 1, 1)="1": FOR a=1 TO 7: e(a)=
    INT(RND*90)+10: NEXT: h=INT(RND*90)+10:
    a=10: b=10: c=2
170 REM***Hoved-LØKKE***
180 IF a=0 THEN 520
190 PEN f(a): FOR d=1 TO INT(a/10): b(d)=10:
    c(d)=10: NEXT: CLS: FOR d=1 TO 20: LOCATE
    d, 1: PRINT CHR$(127): LOCATE d, 20: PRINT
    CHR$(127): LOCATE 1, d: PRINT CHR$(127):
    LOCATE 20, d: PRINT CHR$(127) : NEXT:
    IF LEFT$(a$(a), 1)<>"0" THEN LOCATE
    10, 1: PRINT "mm"
200 IF MID$(a$(a), 2, 1)<>"0" THEN LOCATE
    20, 10: PRINT "m": LOCATE 20, 11: PRINT "m"
210 IF MID$(a$(a), 3, 1)<>"0" THEN LOCATE
    10, 20: PRINT "mm"
220 IF RIGHT$(a$(a), 1)<>"0" THEN LOCATE
    1, 10: PRINT "m": LOCATE 1, 11: PRINT "m"
230 e=0: FOR d=1 TO 7: IF e(d)=a THEN e=1:
    f=d: GOTO 250
240 NEXT
250 LOCATE 1, 22: PEN 7: PRINT "DU HAR"; g;
    "NØGLE";: IF g<>1 THEN PRINT "R"
260 IF i THEN PRINT "DU HAR SKATTEN"
270 REM***Lille LØKKE***
280 IF e THEN LOCATE 10, 10: PEN 2: PRINT
    CHR$(255)
```

```
290 IF a=h THEN PEN 0:LOCATE 9,9:PRINT
    CHR$(254);CHR$(254),CHR$(254):LOCATE 9,
    10:PRINT CHR$(143);CHR$(254);CHR$(143):
    LOCATE 9,11:PRINT CHR$(254);CHR$(254);
    CHR$(254)
300 REM***Joystick-rutine***
310 LOCATE b,c:PRINT "m":d=JOY(0):a$=BIN$
    (d,4):FOR d=1 TO 4:d(d)=VAL(MID$(a$,d,1))
    :NEXT:b=b+d(1)-d(2):c=c+d(3)-d(4):IF
    b=1 THEN 430
320 IF b=20 THEN 450
330 IF c=1 THEN 470
340 IF c=20 THEN 490
350 LOCATE b,c:PEN 6:PRINT CHR$(248):IF
    JOY(0) THEN SOUND 1,1000,10,15,1,1,15
360 REM***Uhyrer, NØGLER, skat***
370 FOR d=1 TO INT(a / 10):LOCATE b(d),c(d):
    PRINT " ":b(d)=b(d)-(b(d)<b AND RND<m)+
    (b(d)>b AND RND<m):c(d)=c(d)-(c(d)<c AND
    RND<m)+(c(d)>c AND RND<m):LOCATE b(d),
    c(d):PEN d:PRINT CHR$(225):SOUND 2,d*100,
    10,15,1,1
380 IF b(d)=b AND c(d)=c THEN 520
390 NEXT:IF e AND b=10 AND c=10 THEN g=
    g+1:e=0:e(f)=0:j=g:LOCATE 1,22:PEN 7:
    PRINT "DU FIK EN NØGLE":GOSUB 600
400 IF g=7 AND i=0 AND a=h AND b=10
    AND c=10 THEN LOCATE 1,23:PEN 7:PRINT
    "DU FIK SKATTEN":j=8:GOSUB 600:
    PRINT "KAN DU FINDE UD?":i=1
410 GOTO 280
420 REM***Nyt rum***
430 IF c<>10 AND c<>11 OR RIGHT$(a$(a),1)
    ="0" THEN b=b+1:GOTO 320
```

```
440 a=a-1:b=19:c=10:GOTO 180
450 IF c<>10 AND c<>11 OR MID$(a$(a),2,1)
    ="0" THEN b=b-1:GOTO 330
460 a=a+1:b=2:c=10:GOTO 180
470 IF b<>10 AND b<>11 OR LEFT$(a$(a),1)=
    "0" THEN c=c+1:GOTO 340
480 a=a-10:b=10:c=19:GOTO 180
490 IF b<>10 AND b<>11 OR MID$(a$(a),3,1)
    ="0" THEN c=c-1:GOTO 330
500 a=a+10:b=10:c=2:GOTO 180
510 REM***Slut***
520 IF a=0 THEN LOCATE 1,23:PEN 7:PRINT
    "DU ER UDE IGEN":j=g+1:IF i THEN PRINT
    "DU HAR SKATTEN MED":j=9
530 IF a=0 THEN GOSUB 600
540 IF a THEN FOR a=0 TO 10:SOUND 4,j(a),
    k(a)*10,7:FOR b=1 TO k(a)*100:NEXT b,a
550 REM Points
560 n=(g*10+i*100-a)*m*100:IF n<0 THEN
    n=0
570 LOCATE 1,25:PEN 7:PRINT n;"POINTS"
580 GOTO 580
590 REM***Musik-subrutine***
600 FOR d=1 TO i(j):SOUND 4,g(d),h(d)*15,7:
    FOR k=1 TO h(d)*150:NEXT k,d:RETURN
610 DATA 119,4,119,1,179,1,119,1,119,8,
    89,4,119,4,179,8,119,4,119,1,179,1,119,
    1,119,8,142,3,159,1,159,12,119,4,119,
    1,179,1,119,1,119,8,89,4,119,4,179,8,
    179,1,179,1,179,4,159,1,159,1,159,4,
    106,2,119,1,89,2,5,8,13,16,21,24,27,30,
    33
620 DATA 119,8,119,8,119,1,119,8,100,4,
    106,4,106,4,119,4,119,8,127,1,119,8
630 SAVE "LABYRINT":GOTO 630
```

10 REM
20 REM
30 MODE 0, blå kant, blå skærm, niveau?
40 Indtastning?
50 Niveau
60 REM
70 Niveau-formel, dimensionering, fanfare, fanfares 9 sektioner, sørgemarch, ENV, ENT
80 Skat, nøgle, hvid kant, farver, kraftigt hvid skærm, labyrint, nordmur
90 Sydmur
100 Vestmur
110 Østmur
120 Rummenes farver, nordlige skillevægge
130 Østlige skillevægge
140 Sydlige skillevægge
150 Vestlige skillevægge
160 Indgang, nøglernes placering, skattens placering, mandens startposition
170 REM
180 Ude?
190 Skillevæggenes farver, uhyrernes placering, slet skærm, tegn skillevægge, nordlig dør?
200 Østlig dør?
210 Sydlig dør?
220 Vestlig dør?
230 Er der en nøgle i rummet?
240 NEXT
250 Skriv med violet, hvor mange nøgler du har
260 Skriv med violet, om du har skatten
270 REM
280 Tegn og mal nøgle
290 Tegn og mal skat
300 REM

310 Slet mand, læs joystick, flyt mand, for langt mod vest?
320 For langt mod øst?
330 For langt mod nord?
340 For langt mod syd?
350 Tegn og mal mand, fodtrin
360 REM
370 Slet uhyrer, flyt uhyrer, tegn og mal uhyrer, uhyrers lyd
380 Spist af uhyre?
390 Fik du nøglen?
400 Fik du skatten?
410 Om igen
420 REM
430 Panden imod den vestlige skillevæg?
440 Gennem den vestlige dør
450 Panden imod den østlige skillevæg?
460 Gennem den østlige dør
470 Panden imod den nordlige skillevæg?
480 Gennem den nordlige dør
490 Panden imod den sydlige skillevæg?
500 Gennem den sydlige dør
510 REM
520 Skatten med ud?
530 Gå til fanfare
540 Spil sørgemarch
550 REM
560 Points
570 Skriv med violet points
580 Frys billedet
590 REM
600 Spil antal sektioner af fanfare
610 Toneværdier til fanfare
620 Toneværdier til sørgemarch
630 SAVE-rutine

a$(109)	Rum
b(9)	Uhyrers plads
c(9)	Uhyrets linje
d(4)	Joystick
e(7)	Nøgler
f(99)	Rummenes farver
g(33)	Fanfarens toneværdier
h(33)	Fanfarens tonelængder
i(9)	Fanfarens sektioner
j(11)	Sørgemarchens toneværdier
k(11)	Sørgemarchens tonelængder
a	Rum
b	Mandens plads
c	Mandens linje
d	Kontrolvariabel
e	Nøglernes flag
f	Den aktuelle nøgles rum
g	Antal fundne nøgler
h	Skattens rum
i	Skattens flag
j	Antal sektioner af fanfaren, som skal spilles
k	Kontrolvariabel
m	Niveau
n	Points

Læg mærke til, at der spilles et antal sektioner af fanfaren svarende til, hvor langt du er nået i spillet. Læg også mærke til labyrinten.

Den er ny fra spil til spil, men den ændrer sig ikke under spillet og er ganske realistisk. Hvis du altså f.eks. kan gå fra rum numer 25 til nummer 35, kan du også gå fra 35 til 25 igen. Hvert rum har en indekseret strengvariabel. F.eks. vil

```
a$(27)="1110"
```

betyde

NØSv

eller døre i nord, syd og øst, men en mur i vest. Tilsvarende kan a$(26) være

1001

Som vi ser kan vi ikke komme mod øst (og forresten heller ikke mod syd). Det er det samme princip, vi brugte i TAXA-spillet.

Hvert rum har også sin særlige farve, som ligeledes forbliver konstant (derfor har vi sættet f(99) til dem) og kan tjene som vejledning. Det er således op til spilleren, om han vil sjusse sig frem eller tegne et kort.

Det sidste kan naturligvis være vanskeligt på de høje niveauer, hvor uhyrerne er hurtige. Der kan hukommelsen blive sat på en hård prøve. Hvor var det nu, skatten var (nu, vi har fået samlet de syv nøgler), og hvilken vej ud?

Programmet er skrevet til joystick, men du kan naturligvis nemt skrive det om til tastatur ud fra, hvad du har lært (kapitel 6). Det er i det hele taget et program, der indeholder det meste af, hvad vi har gennemgået.

Det er jo en slags adventure, og variationsmulighederne er utallige. Du kan lægge mere vægt på tekst og indbygge små logiske opgaver, spilleren skal løse.

Eller du kan udbygge det mere TV-spil-agtige og f.eks. tillade den lille mand at skyde på uhyrerne (brug en rutine hen efter UFO-spillet) og få points for det. Du kan også bruge ideen fra TAXA og lade ham rejse rundt i et rigtigt land. Du skal så inddele kortet over det område, du vil besøge, i nummererede felter. I TAXA har vi f.eks.

```
a$(3)="HUMLEBAEK"
b$(3)="01040500"
```

hvilket giver placeringen

	MØRDRUP
HER ENDER KORTET	HUMLEBAEK ESPERGAERDE
	NIVERØD

```
10 REM***Hoteldetektiv***
20 REM***Startstilling***
30 FOR a=1 TO 5:READ a$(a),b$(a):NEXT
40 b=b+1
50 c=INT(RND*5)+1:IF a(c)=1 THEN 50
60 b(b)=c:a(c)=1:IF b<5 THEN 40
70 d=d+1
80 c=INT(RND*5)+1:IF c(c)=1 THEN 80
90 d(d)= c:c(c)=1:IF d<5 THEN 70
100 a=INT(RND*5)+1:FOR b=1 TO 5:c$(b)=b$
    (b(b)):d$(b)=CHR$(d(b)+64):NEXT:CLS:PRINT
    "HOTELDETEKTIV"
110 PRINT:PRINT "Operasangerinden Canastafire
    overnatter i et billigt kunstnerpensionat. Ommmmm
    mmmorgenen er hendes smykker borte. I det
    samme pensionat bor A, B, C, D og E. Kandu
    ud fra sagens fakta finde tyven? TastENTER for
    nyt faktum."
120 REM***HovedLØKKE***
130 b=INT(RND*5)+1
140 c=INT(RND*5)+1:IF c=b THEN 140
150 e=e+1:PRINT:PRINT "Faktum nummer";e;
    "er: "
160 ON INT(RND*10)+1 GOSUB 230,240,250,
    260,270,280,290,300,310,330:PRINT "Hvem
    er tyven?"
170 a$=INKEY$:IF a$="" THEN 170
180 a$=UPPER$(a$):b=ASC(a$):IF b<65 OR b>
    69 THEN 130
```

```
190 b=b-64 : PRINT : IF b=d(a) THEN PRINT
    "Tyven tilstod"
200 IF b<>d(a) THEN PRINT d$(a);" var tyven"
210 GOTO 210
220 REM***Fakta***
230 PRINT d$(b);" ER " ; a$(b) : RETURN
240 PRINT d$(b);" ER "; c$(b) : RETURN
250 PRINT a$(b);"EN ER "; c$(b) : RETURN
260 PRINT "TYVEN ER " ; a$(a) : RETURN
270 PRINT "TYVEN ER ";c$(a) : RETURN
280 PRINT d$(b);" ER IKKE "; a$(c) : RETURN
290 PRINT d$(b);" ER IKKE "; c$(c) : RETURN
300 PRINT a$(b);"EN ER IKKE "; c$(c) : RETURN
310 IF b=a THEN b=c
320 PRINT "TYVEN ER IKKE "; a$(b) : RETURN
330 IF b=a THEN b=c
340 PRINT "TYVEN ER IKKE "; c$(b) : RETURN
350 DATA DANSKER , FORFATTER , SVENSKER ,
    SKUESPILLER , NORDMAND , OPFINDER ,
    TYSKER , MALER , FRANSKMAND , KOMPONIST
360 SAVE "HOTELDETEKTIV" : GOTO 360
```

```
 10 REM
 20 REM
 30 Nationaliteter og stillinger
 40 Tæller af stillinger
 50 Har vi haft denne stilling?
 60 Sidste stilling?
 70 Tæller af navne
 80 Har vi haft dette navn?
 90 Sidste navn?
100 Tyvens identitet
110 Tekst
120 REM
130 Første variabel
```

140 Anden variabel, har den samme værdi?
150 Tæller af fakta
160 Hvilket faktum?
170 Indtastning?
180 Bud på tyvens identitet?
190 Tilståelse
200 Det rigtige svar
210 Om igen
220 REM
230 Navn-nationalitet
240 Navn-stilling
250 Nationalitet-stilling
260 Tyvens nationalitet
270 Tyvens stilling
280 Navn-nationalitet benægtende
290 Navn-stilling benægtende
300 Nationalitet-stilling benægtende
310 Check variabler
320 Tyvens nationalitet benægtende
330 Check variabler
340 Tyvens stilling benægtende
350 DATA
360 SAVE-rutine

a$(5) Nationaliteter
b$(5) Stillinger
c$(5) Stillinger i forhold til nationaliteter
d$(5) Navne

a(5) Flag (stillinger)
b(5) Relationer (nationalitet-stilling)
c(5) Flag (navne)
d(5) relationer (nationalitet-navn)

a Tyvens identitet
b Kontrolvariabel
c Kontrolvariabel
d Kontrolvariabel
e Kontrolvariabel

HOTELDETEKTIV

Operasangerinden Canastafire overnatter i et billigt kunstnerpensionat. Om morgenen er hendes smykker borte. I det samme pensionat bor A, B, C, D og E. Kan du ud fra sagens fakta finde tyven? Tast ENTER for nyt faktum.

Faktum nummer 1 er:
D ER IKKE TYSKER
Hvem er tyven?

Faktum nummer 2 er:
TYVEN ER SKUESPILLER
hvem er tyven?

Faktum nummer 3 er:
D ER FRANSKMAND
Hvem er tyven?

Faktum nummer 4 er:
NORDMANDEN ER FORFATTER
Hvem er tyven?

Faktum nummer 5 er:
TYVEN ER DANSKER
Hvem er tyven?

Faktum nummer 6 er:
E ER NORDMAND
Hvem er tyven?

Faktum nummer 7 er:
C ER IKKE SVENSKER
Hvem er tyven?

Faktum nummer 8 er:
SVENSKEREN ER IKKE MALER
Hvem er tyven?

Faktum nummer 9 er:
B ER IKKE SVENSKER
Hvem er tyven?

Faktum nummer 10 er:
C ER IKKE MALER
Hvem er tyven?

Faktum nummer 11 er:
TYSKEREN ER MALER
Hvem er tyven? (C)

Tyven tilstod

I HOTELDETEKTIV "blandes kortene" på samme måde som i tal-vende-programmet i kapitel 5. Der vælges tilfældigt, men når et element er "taget", viser dets flag 1, så det ikke kan tages igen. Det er blandt andet dette princip, du bruger, når du giver kort i simulationer af kortspil, og du helst ikke vil have to hjerter es!

Vi leger (og lærer) med andre ord på fuldt tryk. Forfatteren lister ud.

```
10 DIM a(25), b(25), c(25), d(25): a$="11315161911
   2325292132333536393143454941535556585 95":
   FOR a=1 TO 25: a(a)=VAL(MID$(a$, a*2-1,
   1))+5: b(a)=VAL(MID$(a$, a*2, 1))+10: c(a)=
   INT(RND*20)+1: d(a)=INT(RND*20)+1:
   NEXT: MODE 0
20 FOR a=1 TO 25: LOCATE c(a), d(a): PRINT
   "m": c(a)=c(a)+(c(a)>a(a))-(c(a)<a(a)): d(a)=
   d(a)+(d(a)>b(a))-(d(a)<b(a)): LOCATE c(a), d(a):
   PRINT CHR$(143): NEXT: GOTO 20
```

STIKORDSREGISTER

Skrevet af AMSTRAD med programmet på side 94

Amstrad BASIC er skrevet specielt til Amstrad hjemmecomputeren.

Bogen gennemgår programmeringssproget BASIC helt fra grunden.

Samtidig udforskes maskinens muligheder og forfatteren fortæller, hvordan du bedst kan udnytte dens fine egenskaber.

Bogen indeholder en mængde program-eksempler, herunder mange færdige spil og lege, der bidrager til at gøre arbejdet med computeren underholdende og lærerigt.

Bogen henvender sig til såvel begyndere som viderekomne og den er ideel, hvis du vil lære dig selv BASIC.

BORGEN

Bogen kan bruges med CPC 464, CPC 664 og CPC 6128

ISBN 87-418-7530-3

Erwin Neutzsky-Wulff

AMSTRAD BASIC

Borgen